CHARACTERIZING NETWORK INFRASTRUCTURE USING THE DOMAIN NAME SYSTEM

A Dissertation
Presented to
The Academic Faculty

By

Panagiotis Kintis

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of School of Computer Science

Georgia Institute of Technology

December 2020

CHARACTERIZING NETWORK INFRASTRUCTURE USING THE DOMAIN NAME SYSTEM

Approved by:

Dr. Emmanouil Antonakakis
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Douglas Blough
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Angelos Keromytis
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mustaque Ahamad
School of Computer Science
Georgia Institute of Technology

Dr. Jonathan M. Smith
Department of Computer and Information Science
University of Pennsylvania

Date Approved: October 1, 2020

Reason is immortal, all else mortal.

Pythagoras of Samos

To my family and friends, for being there.

ACKNOWLEDGEMENTS

Several people have made the last few years very exciting and productive. I owe a great debt of gratitude to exceptional researchers and devoted friends, who have played a significant role in my personal and professional life, and helped me in many different ways to complete this thesis.

I would like to thank my advisor, Manos Antonakakis, who has been there throughout this journey to guide and assist. He pushed me to achieve my full potential, helped me navigate the research world, and provided me with the intellectual capital to complete this thesis and become a better researcher. Several years ago, he gave me a chance, accepting me in the PhD program, and taught me everything I needed to succeed.

In addition to my advisor, I would like to thank Chaz Lever, a collaborator and, most importantly, a very good friend. He made sure to help me become very good technically, introduced me to a plethora of new technologies, and assisted me in the development of every system used in this thesis. He was there to listen, influence, motivate, and inspire both professionally and personally; for that, and so much more, thank you, Chaz!

In my very first steps in the academic realm, I was fortunate enough to collaborate with very smart people whose influence has been paramount to this work. Dave Dagon, the first person I met at Georgia Tech, introduced me to the DNS world, believed in me, and gave me the chance and tools to work on many interesting problems. Nick Nikiforakis, Michalis Polychronakis, and Roberto Perdisci are three amazing people and outstanding researchers, who always helped me see problems from a different angle. I cannot forget, of course, Angelos Keromytis, who became an integral part of my professional life the last few years. Thank you, everyone, for your invaluable contribution.

My time at Georgia Tech would not have been as enjoyable if it was not for the Astrolavos Lab and its members. Everyone, in their own way, was there for the good and the hard times. I would like to thank Yacin Nadji, Yizheng Chen, Thanasis Kountouras, Omar

Alrawi, Logan O'Hara, Thomas Papastergiou, Thanos Avgetidis, Konstantinos Karakatsanis, Miuyin Yong Wong, Kleanthis Karakolios, Aaron Faulkenberry, William Garrison, Alex Neal, and Michael Mitchel. You helped more than you can imagine, both as collaborators and friends.

I would also like to thank my family in Greece, whose unconditional love and support throughout these years allowed me to work on this thesis. Thousands of miles away, they were always in my thoughts and knowing I was in theirs gave me strength to continue. Everything I have achieved, is because you made sure I could, many years ago. Of course, Niki, who knowing we would be far from each other, never let that be a barrier, but did everything to help and firmly pushed me towards the right direction, no matter what. My other family, away from my family, here in Atlanta. Chris, Lula, Pano, George, words are just not enough; Nikolaki, Vicki, Ismini, Ioanna, Nefeli, you made me feel I belong. You all really gave me a family here, and for that, I will always be grateful.

Finally, I would like to thank my committee members. Doug Blough, the first professor I had the chance to work with, who taught me enough to make the contributions of this thesis a reality. Mustaque Ahamad, who helped me see beyond a single research direction and helped me understand research applicability. Jonathan Smith, who believed in me and our work, helped design a path for this thesis, and introduced me to a research community outside of Georgia Tech. Thank you all for your feedback, guidance, and assistance throughout this process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

From the early 90's until the recent years we have seen a significant amount of protocols and applications being built on top of the Internet Protocol (IP). The ever growing use of off-the-shelf solutions and vertically integrated software is quickly transforming the Internet to an end-to-end encrypted network. This creates a great burden on security applications and the security industry as a whole, which rely on techniques like Deep Packet Inspection (DPI) to secure networks. However, the Domain Name System (DNS), the Internet's phone book, is still available to the security community for both research and applied security. At the same time, DNS monitoring is less invasive, since it is separate from applications using it, preserving the privacy level encryption attempts to set. Hence, DNS is expected to be available to security applications for the foreseeable future and can still be used to reason about the IP even though encryption may make the underlying data unavailable to network security solutions.

This thesis shows how to actively query domain names in order to assist in detecting security threats and provide context around Internet Protocol addresses. Specifically, it introduces the *Active DNS* data, a public dataset that maps almost 70% of the registered domain names to IP addresses from 75% of the Top Level Domains (TLDs) in an active and scalable fashion, as an alternative to extensively used passive DNS datasets. Moreover, this thesis, describes problems faced after operating the Active DNS data generation system for almost five years and how architectural changes improved system availability, reliability, and scalability. Finally, it demonstrates the value in the Active DNS data by performing the first large scale study of *Combosquatting*, an attack technique that utilizes over 2.1M domain names, resolved more than 10B times per day, and attempts to hide malicious activity in at least seven different types of online abuse.

CHAPTER 1
INTRODUCTION

The Internet Protocol (IP) [1] provides the foundation for communication over the Internet. For the last four decades, IP has also become an integral part of almost every application that needs to exchange data with another one across multiple hosts. Apart from the outstanding benefits the Internet has provided in the *Information Age*, it has also become the medium used by miscreants for illicit activities. Securing information and communication systems against adversaries is paramount.

Over the last few decades, we have been basing our network defenses on understanding the IP infrastructure and the way it is being used. Passive collection and monitoring of network traffic has been the cornerstone of our measurements around the IP infrastructure, expressed primarily through Intrusion Detection Systems (IDS) [2, 3, 4, 5, 6, 7]. These systems can take advantage of the context and extensive information around applications communicating and classify communication events as benign or malicious. Thus, it is easy to take action and properly defend a network against malicious activity.

However, the growing volume of traffic on modern networks makes inspection of packets in real time, as well as storage of every communication event, particularly hard. Proposed solutions to this problem include network traffic summaries like NetFlow [8], which can result in a significant reduction of the sheer size of data otherwise collected (e.g., complete network traffic captures) at the expense of losing application information and context. IP traffic summaries are easier to obtain, store, and analyze, compared to network traffic captures, but lack the application layer context that Deep Packet Inspection (DPI) used to provide. Network security solutions that rely on IP summaries are less effective than alternatives that rely on DPI [9, 10]. At the same time, the growing use of encryption in online communication is making DPI more challenging. Implementing encryption standards has

been made easier for developers to protect against illicit eavesdropping, and it is as simple for cybercriminals to also use encryption standards (e.g., symmetric cryptosystems [11], Certificate Pinning [12], etc.) in the same fashion that effectively bypass monitoring systems. The large volume of network data generated on the Internet and the extensive use of encryption make the use of techniques that rely on network traffic visibility (e.g., IDS) more difficult, while DPI is becoming very expensive in large networks. These challenges suggest better techniques are necessary to study IP infrastructure.

The Domain Name System (DNS) [13, 14] is a distributed hierarchical database that acts as the Internet's phone book. The main goal of DNS is to translate human readable domain names to IP addresses that computers require in order to communicate. This thesis shows how to actively query domain names in order to assist in detecting security threats and provide context around Internet Protocol addresses. It attempts to bridge the gab between the information that was becoming available through full packet inspection, and the endured losses from the lack of it, through a novel dataset publicly available to the scientific community and operational researchers.

The first study discussed in this thesis addresses issues revolving around IP intelligence and attempts to ease the burden of data collection, for both research and applied security purposes. It describes a system, `Thales`, which is able to generate large amounts of DNS queries and collect the corresponding responses from the Internet. The system resolves hundreds of millions of domain names, via billions of DNS resolution requests, responses to which provide an adequate view of the infrastructure used on the Internet. After collecting approximately six months of data, we provide empirical evidence which suggest that actively collected DNS data (referred to as *Active DNS*) can be used in several security applications and related research. In fact, we demonstrate that Active DNS data provides much larger breadth, when it comes to identifying portions of the IPv4 and IPv6 address space, compared to passive DNS data collected over the same period of time. However, Active DNS does not provide as much depth as passive DNS. In the context of a bipartite

graph between domain names and IP addresses, Active DNS provides a larger and sparse graph, whereas passive DNS provides a smaller but more dense graph. Finally, we show that Active DNS data can be used in security research and applications, since it includes vital information around domain names and IP addresses, long before we are aware of whether either are used for benign or malicious purposes. Therefore, Active DNS can be a viable alternative source of intelligence to passive DNS. The Active DNS data is made freely available to the security community for research purposes.

The next study focuses on architectural changes and lessons learned from our Active DNS system. After operating `Thales` for more than four years and daily aggregating and providing data to the security community, we discuss problems that we faced with our system architecture and how we solved them by adopting new state-of-the-art big data technologies, as they were becoming available. We detail issues around older technologies that we had been using for instrumentation (e.g., `LXC`, `LXD`, etc.) and data collection (e.g., `tcpdump`, `pcapdump`, etc.), and how they affected the completeness of our collected data. Moreover, we show that appropriate new distributed computing technologies, have made `Thales` more reliable, scalable, and easily distributed. Finally, we show that as the system's robustness and reliability grows, we can extract more DNS related information and better understand how infrastructure on the Internet is being used. After operating the system for years, we evaluate the completeness of our data once again and show that Active DNS data is still valuable and a viable alternative to passive DNS. To date, more than 70 organizations are using Active DNS data for security research related purposes.

The final part of this thesis provides the first large scale study of *Combosquatting*, a technique widely used by miscreants to hide attacks in plain sight. A combosquatting domain name is composed of two parts: (1) a well known trademark or popular domain name, and (2) a mixture of words and characters, prepended, appended, or both, to the "base" domain name (1). For example, given the domain name `gatech.edu`, one could generate and register the combosquatting domain name `gatech-login[.]com`. We

show that most combosquatting domain names lack a generative model, unlike other types of domain squatting (e.g., typosquatting, bitsquatting, etc.), and have a long lifetime, that spans more than a year, inline with similar work conducted in the past [15]. These suggest that combosquatting domains are hard to detect, predict, take down, or defensively register. Moreover, we see very few instances of the domain names in Open Source Intelligence (`OSINT`), which also suggests that combosquatting domains usually go unnoticed. At the same time, we find more than 2.1M combosquatting domain names registered over the course of approximately five years, which have been resolved more than 10B times per day, in several occasions. Lastly, we attempt to understand the way combosquatting domains are used on the Internet, through a passive and active analysis. We collect screenshots and the source code of the websites hosted on combosquatting domain names, and find that combosquatting is used in several different types of abuse, including, but not limited to, phishing, social engineering, affiliate abuse, Advanced Persistent Threats (APT), and trademark abuse (i.e., capitalizing on the popularity of trademarks to sell own products and services).

Through the empirical studies discussed, we demonstrate a system that can provide information adequate to widely used passive DNS data, that can help the security community tackle emerging threats. We also discuss how this system can be replicated and what are the problems that one will face in doing so. After operating `Thales` and providing Active DNS data to the community for almost five years, we discuss architectural changes and their benefits. Finally, our research in combosquatting domain names demonstrates the value in the Active DNS data and how researchers can take advantage of Active DNS to study and understand different phenomena on the Internet.

1.1 Hypothesis

Passive DNS data (1) is expensive to purchase, collect, and store, (2) comes with legal obligations (e.g., anonymization, access control), (3) is local to the monitored network,

and (4) is limited to data points for domains resolved in the network. We hypothesize that actively generated and collected DNS data, in a systematic, scalable, and consistent fashion, can both complement Passive DNS data, and also provide a viable alternative when Passive DNS data is not available, for security research and operational applications.

1.2 Thesis Statement

This thesis shows how actively queried domain names can assist in detecting security threats and provide context around Internet protocol addresses. Specifically, we show how actively collected DNS data (1) provides approximately 200 times more data points than Passive DNS data collected at a large University network, (2) includes information about malicious infrastructure from days to months before the infrastructure is known to be malicious, and (3) enables research into previously unknown attack techniques, like Combosquatting, millions of instances of which can be monitored on a periodic fashion.

1.3 Contributions

Active DNS Measurements: Domain Name System datasets have been used for a long time in security research and security applications [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 15, 29], with passive DNS being the most popular source of intelligence. However, passive DNS data is very expensive to purchase and collect, comes with legal obligations, is only local to the network being monitored, and has limited DNS resources. To make access to DNS data easier, free of charge, and available to the security community and researchers, we design `Thales`, a system that collects DNS data by actively submitting DNS resolution requests for domain names. This study shows that actively collected DNS data, like Active DNS, can be of significant value for security research and security applications.

Active DNS Collection for a Quinquennium: The Active DNS data has been collected and shared with the research community for almost five years. Over this time, we have identified several architectural issues that has led us to better the system that performs

the collection and the way we process, store, and share data. This thesis describes problems with older technologies, new technologies used to replace components of the Active DNS data collection system, and what the future holds. It also demonstrates the added value in the Active DNS data while we adopt new techniques and the system evolves.

Combosquatting Domain Name Threats: Adversaries have been devising several techniques for attacks, which evolve and adapt to new defenses. Combosquatting domain names is an attack technique that relies upon the value and trust users have on legitimate popular brands and domain names. Taking advantage of information the user expects to see (e.g., the word `gmail` when viewing their Google email inbox), adversaries trick users into downloading malicious software, entering credentials in fraudulent websites, purchasing counterfeit products, or even miss an ongoing attack. We thoroughly study the combosquatting technique and shed light in this type of attack. We find that millions of domains, resolved billions of times, are used for at least seven different attack types over almost five years. At the same time, we see an increasing trend in the use of combosquatting domains and lack of detection and mitigation from security systems.

1.4 Dissertation Overview

Chapter 2, provides a background of related material, important in the context of this thesis. It will provide the foundation of the research discussed later, including technical details around DNS, DNS packets, the resolution process, and data collection (Section 2.1). Finally, it concludes with an overview of previous work in the DNS security field (Section 2.2) related to this thesis.

The next chapter, Chapter 3, presents our Active DNS related measurements. It starts by describing `Thales`, the Active DNS data collection system (Section 3.2), and then provides an overview of the collection methodology. It compares the Active DNS to passively collected DNS data (Section 3.3), and shows the value of Active DNS in network security research. The results of the analysis are presented in Section 3.4.

We continue with Chapter 4, which paints a picture of issues we faced after running `Thales` for almost five years. We start by explaining architectural decisions we had made when designing the original system, given technologies that were available. We continue by providing an overview of the current state of the system, technologies it uses at the time of writing, and reasons why we adopted them. The chapter concludes with a comparison between the older system and the capabilities we have achieved with the newer implementation.

Chapter 5, provides an empirical study of the Combosquatting security problem. It starts by defining the problem and providing the methodology used to study combosquatting. We measure the extent of the problem, using passive and active DNS data, and perform a lexical and network based analysis around combosquatting domain names. We, then, demonstrate the ways combosquatting domains are used on the Internet for various attack types, and conclude with a discussion around potential remedies.

Lastly, Chapter 6, concludes this thesis, with a summary of the findings presented previously and discusses our limitations.

CHAPTER 2

BACKGROUND

2.1 The Domain Name System

The Domain Name System (DNS) is a core component of the Internet, used since the very early days, and is responsible of translating human readable and memorable domain names to computer-understandable Internet Protocol (IP) addresses. Similar to the Internet's phone book, it is far easier for a user to recall the domain name `www.example.com`, rather than the complex IP address `93.184.216.34`. DNS's responsibility, is to retrieve an IP address for the domain name provided by a user, or system, namely, the *stub resolver*, or simply *stub*.

2.1.1 Domain Names

For DNS to work efficiently and reliably, it takes a hierarchical approach when a resolution occurs. Before we dive deeper into the resolution process, we need to first understand how a domain name is formed. Based on RFC 1034 [30], a domain name consists of *labels*, linked together by a `null` value. For presentation purposes, we visualize the empty value with a `dot`. Every label in the domain name can be up to 63 characters in length and must follow the rules for ARPANET host names [31]. Figure 2.1 depicts the four parts that compose a Fully Qualified Domain Name (FQDN). Even though we read a domain from left to right, when it comes to DNS and resolutions, we follow a right-to-left approach. We can see that the four parts that comprise a domain name are: (1) the *root* (light blue), (2) the *Top Level Domain* (blue) (TLD), (3) the *domain name label* (green), and (4) the *subdomain*, or subdomains that might follow (red).

The root refers to the *root DNS servers*, which are responsible of knowing "where" the

Figure 2.1: Distinct parts of a domain name.

Top Level Domain servers are on the Internet. In other words, they will provide a client with the IP address of the next part of the domain name. In our particular example, the roots will be consulted about the IP address of the `com` TLD. The dot at the very end of domain names is usually implied and therefore it very rarely appears in writing.

There are 1,514 TLDs, at the time of writing, each one responsible to know the IP address of the host that is *authoritative* for a domain name that follows. In this example, the `com` TLD will know the IP address of the server that has information about `example.com`. Hence, when consulted, the TLD will provide this information to the requester.

Finally, the `example` part of the FQDN, which we refer to as the domain name label (also, label, registered portion of the domain name, etc.), is tied to the server responsible for everything under the `example.com` domain. Therefore, when asked, this server is able to respond with the IP address that the domain name `www.example.com` points to.

2.1.2 Domain Resolution

The way domain names are formed help towards the resolution process of a domain name. One thing that we have not discussed yet, is the notion of DNS *zones*. A zone, in DNS, refers to the smallest stand-alone delegated part of domain names. DNS is not only hierarchical, but also highly distributed. For the distributed system to work, certain pieces are delegated to different entities. As hinted earlier, while we traverse a domain name from right to left, we can identify certain entities that are responsible for entities "under them".

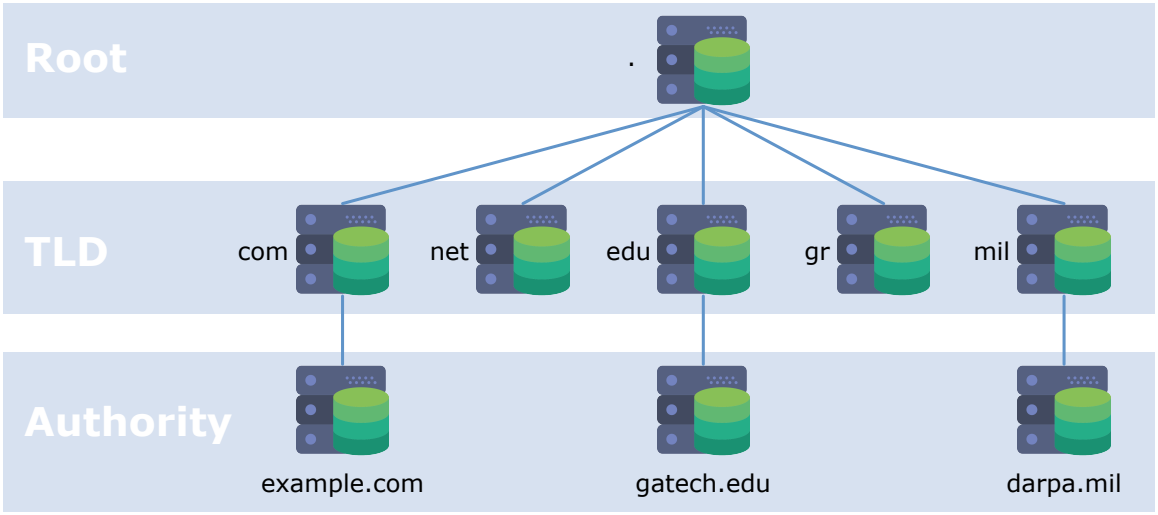


Figure 2.2: Example of the DNS hierarchy.

Figure 2.2 depicts the hierarchy in DNS. The light blue boxes grouping levels together represent different types of zones. For instance, we can see the root zone at the very top, followed by TLD zones, which are next followed by the `authoritative` zones. An authoritative zone is the most specific delegation in the hierarchy. For example, in this particular figure, the `example.com` zone has been delegated to the `authoritative nameserver` labeled as "example.com". The term authoritative nameserver is often shortened to `nameserver`, `ns`, or `authority`. These terms are used interchangeably in this thesis, and they all refer to an authoritative nameserver, responsible for a particular zone, unless otherwise specified.

Resolving a domain name refers to the process of retrieving the IP address that a particular domain name "points to". For instance, `www.example.com` points to `93.184.216.34`. This information is stored in the authoritative nameserver of the registered domain `example.com`. However, a problem that is immediately apparent, is that we have no idea how to contact `example.com` in order to "ask" for this IP address. The hierarchical model in DNS allows for a tree traversal from top to bottom, discovering more and more information as we ask. This process is taken care of, for the system, by a `recursive DNS resolver` (also referred to as recursive server, or recursive — terms used interchangeably

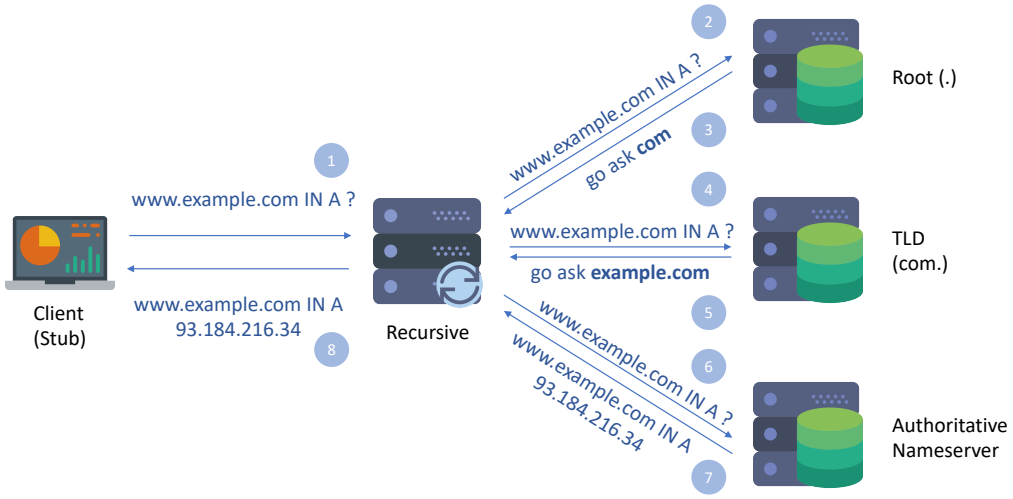


Figure 2.3: Domain name resolution process.

in this thesis and all refer to a recursive DNS resolver, unless otherwise specified).

Figure 2.3 shows the process of resolving the domain name `www.example.com` for a client "Client". At the very fist step (1), the client asks the recursive server "Recursive" about the IP address (`A` record, see 2.1.3) of the FQDN `www.example.com`. Assuming that the recursive does not know the answer already (see step (8)), it will first ask the root server whether it knows the IP address of the provided FQDN (step (2)). The root server will (most likely) not have the answer, therefore redirecting the recursive to the `com` TLD (3). The recursive will then forward the same query to the TLD server (4), which will also (most likely) not have the response ready. The TLD, will then redirect the recursive to the authority for the `example.com` zone (5). Next the recursive, will ask the authority (6) for the FQDN. Provided that there is no further delegation for `www` under the `example.com` zone, the authority will provide an authoritative response, which will contain the IP address of the FQDN (7). In that response, the authority will also hint the recursive towards the longevity of that domain to IP association, using a specific field called *Time-To-Live* (TTL). The recursive will save the response in its local cache (and evicted TTL seconds later) and

then forward the response to the client, in step 8.

Throughout this process several DNS packets will traverse the Internet and will be exchanged between multiples entities. Usually, the IP addresses of popular TLDs and authorities will be in the local cache of the recursive, hence reducing the time it takes to retrieve a response. Overall, the whole process will take approximately a couple to 100 milliseconds, with several exceptions. The default timeout for the process is three seconds, after which the recursive might retry, or reply back to the client with message stating the problem.

2.1.3 DNS Packets & Contents

During the resolution of a domain name, at least two DNS packets are involved; one that goes to the recursive from the client, and one that carries the response from the recursive back to the client. At this point, we should note that DNS by default works over the UDP protocol, with datagrams of up to 512 bytes in size. In case UDP is unavailable, as a fallback mechanism, or in case a response is larger than 512 bytes, DNS can also run over TCP. The default port that is used by DNS authorities and recursives is 53. That is effectively the port they listen for requests coming in to. However, DNS requests are submitted over a higher ephemeral port. For security purposes, and to avoid *DNS cache poisoning attacks* [32, 33, 34], DNS queries will leave through a random port, different for each outgoing query. The incoming response must be received by the correct port for it to be validated and accepted by the client. This is an important characteristic that could present a bottleneck when submitting DNS requests at scale. Chapter 3, Section 3.2.1, explains how we work around this issue in our Active DNS work.

Figure 2.4, shows an example DNS response to the client from a recursive server. We can clearly see the FQDN `www.example.com` being queried by the stub at the very top of the figure. The rest of the packet is composed of five main components.

DNS Headers

First, the DNS headers describe technical information around the response and the packet itself to assist in interpreting the content:

- `OPCODE`: A four bit field that specified the kind of the query in the packet. This is set by the requester (the client in this case) and is repeated in the response.
- `RCODE`: Stands for Response Code and is visualized with the word `STATUS` in the figure. It denotes whether there was an issue with the query. There are several different `RCODE`s, which include, but are not limited to:
  - `NOERROR` (0): the transaction should be considered successful.
  - `FORMERR` (1): the DNS query was not formed properly.

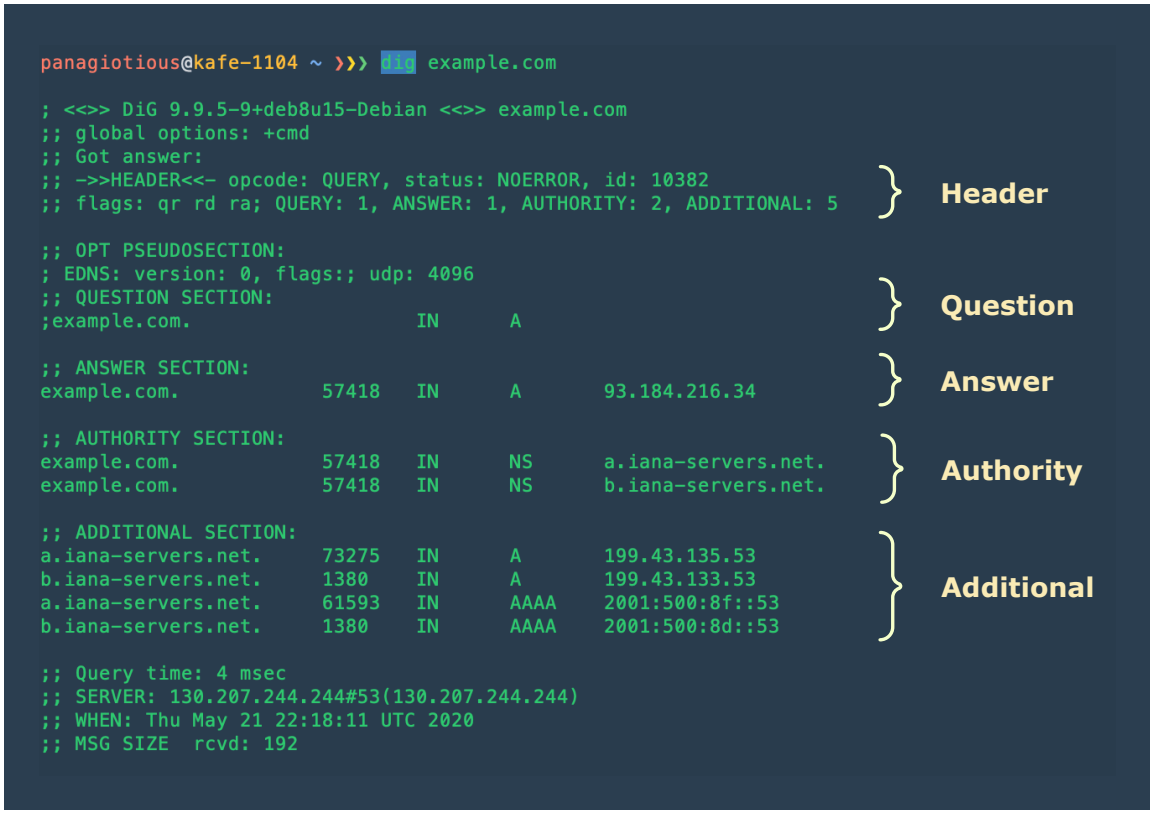


Figure 2.4: A DNS response from the `dig` command line tool.

  - SERVFAIL (2): some error occurred that prevented the server from successfully responding.
  - NXDOMAIN (3): the requested domain does not exist.
  - REFUSED (5): the server will not respond to this client; for example the client is not authorized to use the particular DNS server.

- TXID: The transaction ID, depicted as id in the figure, which is a unique random number that was generated by the client and sent back to it from the server to verify the authenticity of the response.

- Flags: A sequence of flags that describe the packet, including but not limited to:
  - AA: The packet is an Authoritative Answer from an authority that is authorized to respond for the requested zone.
  - TC: When set, the packet has been truncated, because it did not fit in 512 bytes; the client should retry over TCP.
  - RD: The client is asking the recursive to pursue the query recursively, until it receives a response. An alternative to recursive resolution is the iterative one, during which a server without an authoritative answer (AA) will reply with a reference to a server who might have the answer. On the right-hand side of the resolution process in Figure 2.3, the queries sent between the recursive and the DNS hierarchy, are iterative queries.
  - RA: The DNS server supports recursion. Stands for Recursion Available, and notifies the client that it will perform a recursive resolution to identify the answer to a question.

- QUERY: The number of *questions* in this packet.

- ANSWER: The number of *authoritative answers* in this packet.

- `AUTHORITY`: The number of *authority* entries in this packet.
- `ADDITIONAL`: The number of *additional* entries in this packet.

Resource Records

The rest of the packet is formed with a sequence of *Resource Records* (RRs). A Resource Record is a complete piece of information in DNS. It consists of five items, as seen in Figure 2.5. The leftmost item, `example.com`, is the *name* that is being queried or resolved. This is commonly referred to as the `QNAME` (query name) or the `RNAME` (response name), depending on whether it is in the *question* or *response* section, respectively. More about these two sections later. In this thesis, the term `qname` will be used several times and it always refers to an FQDN, whether that is a `QNAME`, or an `RNAME`, unless specified otherwise. The term QNAME is very often used as an abbreviation of the term "domain name", because it is the domain that a client wishes to resolve.

The second item is the TTL, or how long this RR is valid for, in terms of caching for the recursive and the client, as discussed earlier, in Section 2.1.2. This is an integer that represents time in seconds.

The third item is the RR `CLASS` code. The class defines the kind of the record; for example, `IN` stands for Internet, which hints that this RR is an Internet related RR. Other classes include `CS` for the `CSNET` class (obsolete), `CH` for the `CHAOS` class queries, related to Chaosnet [35], and `HS`, for Hesiod, a name service part of *Project Athena* [36].

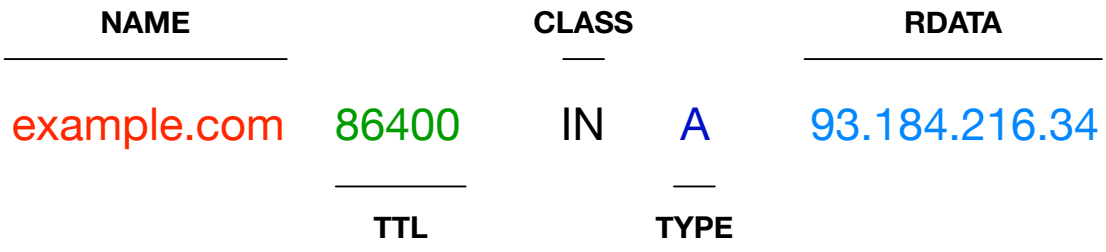


Figure 2.5: The format of a Resource Record (RR).

The fourth item in the RR is the TYPE. Similar to the NAME, depending on whether it is in the query or response section, we will see the abbreviations QTYPE for query type and RTYPE for response type, respectively. Like the term QNAME, in this thesis, the term qtype is used to represent the type of a DNS record, irrespective of the section of the packet, unless otherwise specified. Hence, the terms qtype and rtype can be used interchangeably. There are more than 100 different qtypes used in DNS, each of which has a specific role and provide context to the RDATA that follows. That is, when the qtype is equal to A (1), we know that the RDATA must be an IPv4 address. When it is equal to AAAA (28), we know that the rdata must be an IPv6 address. Other popular qtypes include, but are not limited to:

- NS (2) — Nameserver: an FQDN that defines the authority responsible for the queried zone.
- CNAME (5) — Canonical Name: an alias to a different FQDN.
- SOA (6) — Start of Authority: a seven-field rdata structure that defines information about a particular zone.
- PTR (12) — Pointer: a pointer to a canonical name, but instead of continuing with the resolution, the server will simply return the name; very often used in reverse lookups, when the FQDN that points to an IP address is retrieved.
- MX (15) — Mail Exchange: a two-field rdata structure that includes a priority number and an FQDN to the email exchange responsible for receiving email messages on behalf of the QNAME.
- TXT (16) — Text: an arbitrary string.
- SRV (33) — Service Location: the FQDN for a particular service described in the QNAME; similar to the MX record, but for other protocols.

- `AXFR` (252) — Zone Transfer: asks the authority to exchange all zone information for the specified zone; effectively return the zonefile for the zone.
- `ANY` or `*` (255) — Everything: asks the authority to return any RR it has in its cache.

The final part of an RR (rightmost in Figure 2.5) is the `RDATA`. The RDATA, or Response Data, is the data that the client will eventually receive. In order to understand the context of it, we have to look at the RTYPE for the particular RR. As mentioned earlier, when the RTYPE is, for example, `A`, the RDATA will be an IPv4 address. However, when the RTYPE is `MX`, we expect to find a more complex data structure in the RDATA, which will include both a priority number, and the FQDN for the email exchange service for that particular domain. For example, in Listing 2.1, a client that receives these two RRs should use the one with priority `10` as the email exchange to send an email to `[user]@kael.pw`. Here, the command line utility `dig`, takes care of visually representing the RDATA in a meaningful way for the user, although the actual contents of the packet are slightly different.

```
1 $ dig MX kael.pw
2
3 [...]
4 sink.kael.pw.   0   IN  MX  10  mx1.kael.pw.
5 sink.kael.pw.   0   IN  MX  20  mx2.kael.pw.
6 [...]
```

Listing 2.1: DNS Response

Response Sections

Finally, the RRs we discussed earlier are organized in four different sections in the packet. Each section is related to a part of the communication events that took place throughout a DNS transaction.

The first section is the `QUESTION SECTION`, which includes information about the domain(s) that were queried. As mentioned earlier, cases where more than one question is ever asked are extremely rare, hence, the Question Section will almost always include just one RR. The RR in the Question Section will have a domain name (the QNAME), the query class (`QCLASS`), which is most often `IN` (for Internet), and the query type (QTYPE).

The `ANSWER SECTION`, which immediately follows, will contain any response related RRs. These RRs are similar to Figure 2.5, and will include the RDATA that the client needed in the first place. In the example in Figure 2.4, we can see that the client is provided a single RR, which states that `example.com` can be found on the server with IP address `93.184.216.34`. Multiple RRs could be encountered, similar to Listing 2.1. In that case, the client can choose to use whichever RR it wishes, unless specified otherwise by RFC 1035 [31] (like in MX records).

The `AUTHORITY SECTION` of the packet includes the domain names of the authoritative nameserver of the QNAME being resolved.It is often omitted by the recursive when the answer is created for the stub. For example, in Figure 2.4, we can see that the authorities for `example.com` are both `a.iana-servers.net` and `b.iana-servers.net`. We can tell that these are the authorities because of the `NS` RTYPE in the RRs in the Authority Section.

Lastly, the `ADDITIONAL SECTION`, and last section of the packet, again omitted by the recursive oftentimes, includes the IP address(es) of the RDATA for each RR in the Authority Section. We can see the `A` and `AAAA` RTYPEs in the RRs in the Additional Section, which denote an IPv4 and IPv6 address respectively. Moreover, we know that these are the authorities for the `example.com` zone, since the RNAMEs in the `ADDITIONAL`

SECTION, are the RDATA in the AUTHORITY SECTION. This part of the packet is also known as the *Glue* and the RRs as *Glue Records*. These records come directly from the TLDs (or the authority responsible for the parent zone in general), and help break cycles in resolutions. Although not mandatory to be set by parent zones, Listing 2.2 shows an example of a cycle where resolution would have been impossible if the TLD was not aware of the IP address of ns1.rylai.pw and ns2.rylai.pw.

```
$dig www.rylai.pw

[...]
;; ANSWER SECTION:
www.rylai.pw.           0       IN      CNAME   sink1.rylai.pw.
abcbabd10ffffe73734870e0c4.sink1.rylai.pw. 0 IN A 143.215.215.212

;; AUTHORITY SECTION:
rylai.pw.               3595    IN      NS      ns2.rylai.pw.
rylai.pw.               3595    IN      NS      ns1.rylai.pw.

;; ADDITIONAL SECTION:
ns1.rylai.pw.           3595    IN      A       143.215.215.211
ns2.rylai.pw.           3595    IN      A       143.215.215.211
[...]
```

Listing 2.2: DNS Response

To make the issue clear, one would need to consider one small step that we omitted in Section 2.1.2 during the resolution process. When the recursive is given the "location" of the authority responsible for a domain name (e.g., rylai.pw), it is actually given the domain name of the authority (e.g., ns1.rylai.pw). Then the recursive will try to resolve

that domain name (`ns1.rylai.pw`) and effectively retrieve the IP address of the authority. Glue can help speed up this process, if the TLD already knows the IP address of the authority and simply append it to the packet it replies with to the recursive at step 5 in Figure 2.3. Assuming that the TLD did not have the IP address of the authority readily available, it would only reply with the NS record, hence just providing the recursive with a new domain name, which would need to be resolve in order to get the IP address of the authority. In the case of Listing 2.2, the TLD will reply to the recursive with `rylai.pw $TTL IN NS ns1.rylai.pw.` and `rylai.pw $TTL IN NS ns2.rylai.pw.`. Keep in mind that the question which initiated the whole process was *what is the IP address of `www.rylai.pw`?*. So at this point, the recursive is asking about `rylai.pw` and is told that the entity who knows where `www.rylai.pw` is `ns1.rylai.pw`. When the recursive starts a resolution process for `ns1.rylai.pw`, it will get stuck by receiving the same response from the TLD, or `rylai.pw $TTL IN NS ns1.rylai.pw.`, since all `wwww`, `ns1`, and `ns2` are under the same zone (`rylai.pw`). Therefore, the recursive will not be able to resolve the domain name. Glue will solve this cyclical problem. The TLD "hints" the recursive about the IP addresses of the authorities (or more specifically, the IP addresses the authority domains point to), even when those authority domains are under the same zone as the QNAME being resolved. The domain names that have their authorities' domain names under the same zone that they serve (e.g., the case of `rylai.pw`), are referred to as *tucked* zones or domains. In the work from Kintis, et al [37], you can see an even more important problem that DNSSEC [38] introduces, since it does not promote the Authority and Additional Sections to authoritative, hence, not cryptographically signing them, which means that the recursive has to ignore them. That makes resolving domains like this practically impossible under DNSSEC. We will not go any deeper into DNSSEC, as it is out of the scope of this thesis.

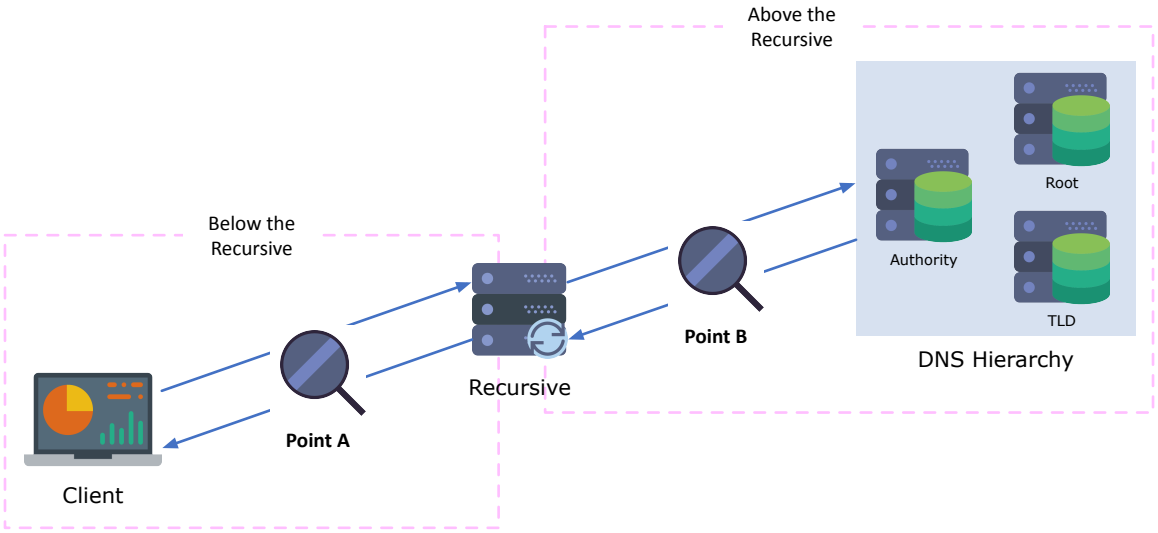


Figure 2.6: Collection points of DNS data in a network around the recursive.

2.1.4 DNS Data Collection

Collection of DNS data has been in the spotlight of the security community for more than a decade [39]. Data collected on the network passively is referred to as *Passive DNS* data. There are several application that utilize DNS data, as we discuss in Chapter 3. Before we go any further in talking about collected DNS data, however, we need to explain the approaches in collection and differences that lie in each one.

Figure 2.6, demonstrates two of the most popular collection points of Passive DNS data: *below the recursive* (Point A) and *above the recursive* (Point B). There are a few significant differences in each approach that we should note. Data collected below the recursive, or at Point A, will include every single request that reaches a recursive from a local (or remote) network that utilizes said recursive. In other words, every DNS packet that leaves a network and reaches the recursive will be visible. That has some unique properties, like the ability to identify the IP address of the client that requests the resolution and receives a response, or the fact that every single resolution request will be visible. On the other hand, data collected above the recursive will not provide the IP address of the client that performs a resolution request, nor will requests that are being replied to by the

local recursive cache will be visible. However, above the recursive is a good vantage point if we are interested in the DNS hierarchy and information about which authority replies for which domain name, information that is not available below the recursive. As one can easily realize, data collection above the recursive is going to be much more efficient and easy, since far less requests will be visible, given TTLs that will keep domains in the recursive's cache.

2.2 Previous Work

2.2.1 DNS Measurements

The collection of passive DNS data has been proposed by Weimer et al. [39] over a decade ago as a method that network operators could use to investigate security events in their environments. Zdrnja et al. [21] was the first to discuss how passive DNS data can be used for spotting security incidents using domain names. Notos [40] and Exposure [41] used the idea of building passive DNS reputation by statistically modeling various properties of the successfully resolved passive DNS traffic. Plonka et al. [42] introduced Treetop, a scalable way to manage a growing collection of passive DNS data and at the same time correlate zone and network properties. Since then, several researchers were able to use proprietary passive DNS data to build systems that can detect abuse in the Internet [17, 18, 43, 19, 44, 45]. Clearly, passive DNS is considered to be a very valuable tool that network operators and security researchers use in the fight against Internet abuse. As already discussed, our active DNS project can provide researchers open access to DNS datasets, comparable to the very useful passive DNS, but without any concerns on personally identifiable information (PII) or other legal barriers to repeatable DNS research.

There have been many commercial and nation efforts to create passive DNS repositories. The costs for the commercial offerings [1] often pose a barrier for researchers and network operators. Now, some of the national efforts are hindered by DNS policy, and

[1] For example, `https://www.farsightsecurity.com/`

thus have yet to be widely adapted by the community. Perhaps the most successful has been `passiveDNS.cn`, which was quickly dismissed as an unreliable source of DNS information. The reason behind this development is very simple. The Chinese operators [2] passively collected DNS records that have been already censored by their egress sensors. In our project, we do not censor the views of the recursive DNS servers that Thales uses to resolve the seed domain names on a daily basis.

With the respect of active scanning efforts, most of the efforts have been conducted from the side of the industry. In the last year, however, new work surfaced from the academic community [25] that provides the ability to researchers to scan the entire IPv4 space and use the results for open security research. This is the work that is closest to the proposed system. The key difference, however, is that Censys was not designed to scan the domain name space, rather, IPv4. Thus, while researchers could find some DNS logs into this great public project, our work both complements Censys and also is designed to deal with DNS scanning.

2.2.2 DNS Abuse

Weimer et al. [39] proposed collecting passive DNS data for security analysis. Since then, researchers have used passive DNS data to build domain name reputation systems using statistical modeling methods to detect abuse on the Internet [16, 41, 17, 18, 43, 19, 44, 45]. More recently, Lever et al. [23] used passive DNS to identify potential domain ownership changes. Hao et al. [46] uses only registration features to build domain reputation system. Liu et al. [47] revealed that dangling DNS records pointing to invalid resources can be easily manipulated for domain highjacking. Chen et al. [24] used passive DNS data to estimate the financial abuse of advertising ecosystem by a large botnet.

[2] `http://www1.cnnic.cn/ScientificResearch/LeadingEdge/fymly1/`

2.2.3 DNS Squatting Abuse

Several studies have focused on domain squatting in general. Jakobsson et al. [48, 49] proposed techniques for identifying typosquatting and discovered that websites in categories with higher PPC ad prices face more typosquatting registrations. Wang et al. [50] proposed models for the generation of typosquatting domains from authoritative ones. Agten et al. [51] studied typosquatting using crawled data over a period of seven months finding, among others, that few trademark owners protect themselves by defensively registering typosquatting domains. In addition to typosquatting, Nikiforakis et al. [52] quantify the extent to which attackers are leveraging bitsquatting [53], where random bit-errors occurring in the memory of commodity hardware can redirect Internet traffic to attacker-controlled domains. Their experiments show that new bitsquatting domains are registered daily and monetized through ads, affiliate programs and even malware installations. The authors later performed a measurement of the so-called "soundsquatting", where attackers abuse homophones to attract users and confuse text-to-speech systems [54].

The only work on combosquatting other than this paper is a brief 2008 industry whitepaper [55]. Starting with 30 trademarks and up to 50 generic keywords the authors constructed possible combosquatting domains and then attempted to get traffic data for the 500 domains that were registered. The authors found that most sites were filled with ads, thereby abusing the popularity of trademarks and diluting their revenue. Motivated by the findings of that nine-year-old whitepaper, we performed the experiments described in this paper finding millions of combosquatting domains and analyzing registration and abuse trends over almost six years.

CHAPTER 3

ACTIVE DNS MEASUREMENTS

3.1 Introduction

The Domain Name System (DNS) is a fundamental component of the Internet. Most network communication on the Internet starts with a DNS lookup that maps a domain name to a corresponding set of IP addresses. Cyber criminals frequently leverage DNS to provide high levels of network *agility* for their illicit operations. For example, most malware relies on DNS to locate its command-and-control (C&C) servers. Such servers are used to send commands from the attacker, exfiltrate secret information, and send malware updates.

DNS abuse is an enduring, if not permanent, feature of the Internet, which might at best be managed through various policies, remediation technologies and defenses. Traditionally, network operators have relied on *static blacklists* to detect and block DNS queries to malware domains. Unfortunately, static blacklists, which are often manually compiled, cannot keep pace with the quantity of network agility of modern threats. This results in blacklists that are incomplete and become outdated quickly.

To overcome the limitations of static blacklists, new analytical systems have been proposed [40, 41, 17, 18, 43, 42] to shorten the response time necessary to react to new threats and secure networks. Those systems rely on the efficient collection and presentation of passive DNS datasets. However, such datasets are difficult to find, challenging to collect, and often require restrictive legal agreements. These obstacles make further innovation difficult and are an impediment to repeatability of research.

The lack of open and freely available DNS datasets puts the security community at a disadvantage because they lack access to datasets describing a critical component used by adversaries on the Internet. Clearly, the security community is in need of open, freely

available DNS datasets than can help increase the situational awareness around modern threats. This is illustrated by the fact that most modern threats rely on DNS for their illicit activities.

This chapter provides a solution aimed at filling this gap. We introduce the concept of active DNS and discuss a new large scale system, Thales, which is able to systematically query and collect large volumes of active DNS data. The output of this system is a distilled dataset that can be easily used by the security community. Thales has been reliably active for more than six months and collected many terabytes of DNS data, while causing only a handful of abuse complaints. Access to this dataset is currently available to the community from the following project website: `https://www.activednsproject.org/`.

3.1.1 Contributions

We present a system, Thales, that can reliably query, collect, and distill active DNS datasets. Due to the public nature of our seed data, our active DNS datasets do not contain any potentially sensitive information that preclude their use by the security community. Thales has been collecting active DNS data for more than six months with almost zero down time (only three days). During this time, the system has generated more than a terabyte of unprocessed DNS PCAPs along with tens of gigabytes of de-duplicated DNS records per day. Thus, the active DNS datasets represent a significant portion of the world's daily DNS delegation hierarchy.

We provide in-depth comparison between the newly collected active DNS datasets and passive DNS collected from a large university network. We show that the active DNS datasets provide greater breadth (i.e., reaches out to a larger portion of the IPv4, IPv6, and DNS space). Conversely, passive DNS yields a denser graph between the queried domain names and the remaining IP and DNS infrastructure.

We practically explore how active DNS can be used to improve the security of modern networks through several case studies. We show that the active DNS datasets can be use for

early detection of financial and other Internet threats. Our analysis shows that more than 75% of malicious domain names appear in the active DNS datasets months before they get listed in a public blacklist. We demonstrate how active DNS can be used to implement and extend existing DNS related research, specifically, by implementing an algorithm used to detected potential domain ownership changes. Finally, we show how active DNS can be used as a signal to identify malicious campaigns on the Internet.

3.2 Active DNS Data Collection

With this section we introduce Thales. We will begin by discussing the network and system infrastructure necessary to systematically and reliably collect the active DNS datasets. Then, we will discuss the details of the domain names that compile the daily seed for Thales. The section will be concluded by discussing the long term measurement behind the collected active DNS datasets.

3.2.1 Infrastructure

The reliable collection of DNS data is far from easy. Thales was designed to retain high levels of availability, efficiency and scalability. The goal of Thales is clear; the generation of active DNS datasets that will provide systematic snapshots of the DNS infrastructure, several times per day. These datasets will enable the security community to construct a timeline of the evolution of threats in the broader Internet.

Our system, Thales, is composed of two main modules as seen in Figure 3.1: (a) the traffic generator and (b) the data collector. The first is responsible for generating large numbers of DNS queries using a list of seed domain names as an input to the system. The second module is responsible for collecting the network traffic and guiding these raw DNS datasets for further processing (i.e., data deduplication).

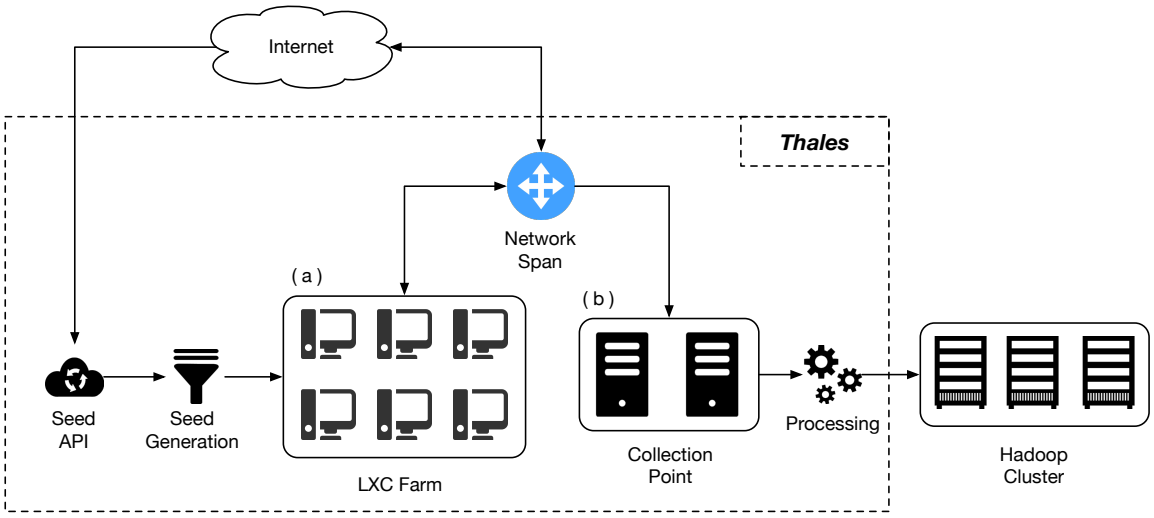


Figure 3.1: The Seed API is responsible for collecting the seed domains from various sources and the Seed Generation reduces them to a list of unique domains. The LXC Farm corresponds to the query generator which is connected to the internet through a Network Span. That in turn is sending traffic to the Collection Point from where data is being reduced and stored for long term on our Hadoop Cluster.

Traffic Generation

In order to achieve high availability, redundant systems are used to generate traffic. Linux containers (LXC) [56] are setup across several physical systems, creating a DNS scanning cluster of 30 LXC containers. Each LXC contains its own local recursive software [1] and is assigned a job, where a subset of the overall daily seed domain names will have to be resolved by a particular container. High efficiency is achieved by increasing the rate of DNS resolution requests (a.k.a. queries per second) that can be handled by the recursive in the LXC container. However, just increasing the resources of the LXC container will not suffice for the container to handle a large enough number of DNS requests. This is because the local recursive in the LXC is bounded by the maximum number of ports that can be used for UDP sockets. This means that the number of requests that can be sent by a host have to be limited to the number of available concurrent ports that the local recursive (in the LXC container) can handle.

[1] We used the Unbound (`https://www.unbound.net/`) recursive software in every LXC container.

At any given point in time, a container could theoretically handle up to 64,512 ($2^{15} - 1024$) sockets per IP address – and therefore 64,512 UDP query packets in transit. The LXC containers support custom network interfaces, which support assigning a different IP address to each container. More specifically, we use 30 contiguous IPs out of an assigned IP block of 63 available addresses (/26). Thus, they are able to send and receive up to $30 \times 64,512 \approx 2^{21}$ simultaneous DNS resolution requests from the infrastructure. These results are achieved by deploying the containers on two physical systems. Each of these two systems has 64 processing cores and 164GB of RAM. It is worth pointing out that using LXC containers allows us to scale the infrastructure horizontally by simply adding more systems to our scanning cluster.

Data Collection

The requests submitted by Thales are collected at two vantage points. The first one is on the LXC container that has submitted the resolution request for a given domain name, whereas the second one is at the SPAN of a switch that routes traffic for all our containers. As mentioned earlier, we are utilizing several IP addresses from several local virtual LANs (VLAN). These VLANs have been "trunked" to a single 1Gbit interface on a host that collects all port 53 UDP traffic. We are collecting traffic at both points for redundancy and verification of correctness for the daily active DNS datasets.

Capturing network traffic results (on average) in a massive 1.67TB of raw data in *packet capture* format (pcap). This data is transferred in a local Hadoop cluster composed of 22 data nodes. The Hadoop cluster is responsible for parsing the pcap files, deduplicating the resource records (RRs) and converting the RRs into meaningful DNS tuples of following format: `(date, QNAME, QTYPE, RDATA, TTL, authorities, count)` as seen in Figure 3.2. Deduplication is a critical step, since many responses we collect remain the same throughout a day. Thus, after removing duplicate RRs, we are left (on average) with approximately 85GB of data per day. Detailed measurements for both daily

```
{
    "date": "20160303",
    "qname": "0746jiaoyou.com.",
    "qtype": 1,
    "rdata": "61.151.239.202",
    "ttl": 3600,
    "authority_ips": "58.216.26.232,120.52.19.142",
    "count": 5,
    "hours": 32778,
    "sensor": "active-dns"
}
```


Figure 3.2: A sample record from our dataset that shows the data fields that are stored. The `authority ips` field represents the authoritative nameservers that replied for this domain name and the `hours` variable captures the hour of the day that this record was seen in a 24 bit integer.

raw and deduplicated RRs will be discussed in Section 3.2.3.

3.2.2 Domain Seed

Before Thales can begin scanning the domain name system, it has to be provided with a list of domain names that will act as candidates for resolutions. We will refer to these domain names as the *seed* for Thales. The seed is an aggregation of publicly accessible sources of domain names and URLs that we have been collecting for several years. These include but are not limited to Public Blacklists, the Alexa list, the Common Crawl project, and various Top Level Domain (TLD) zone files.

More specifically, we are using the zone files that are published daily by the administrators of the zones for com, net, biz and org. In Figure 3.3 we present the number of domains obtained by each zone file. Because of the relative number of small daily changes, compared to the size of the zone files, the daily changes are not that apparent in Figure 3.3. We note that the number of domains obtained by zone files changes as new domains get registered and old ones expire (and get removed from the zone). In Thales we input these zone files that we collect daily to our domain seed. This way our seed includes the current

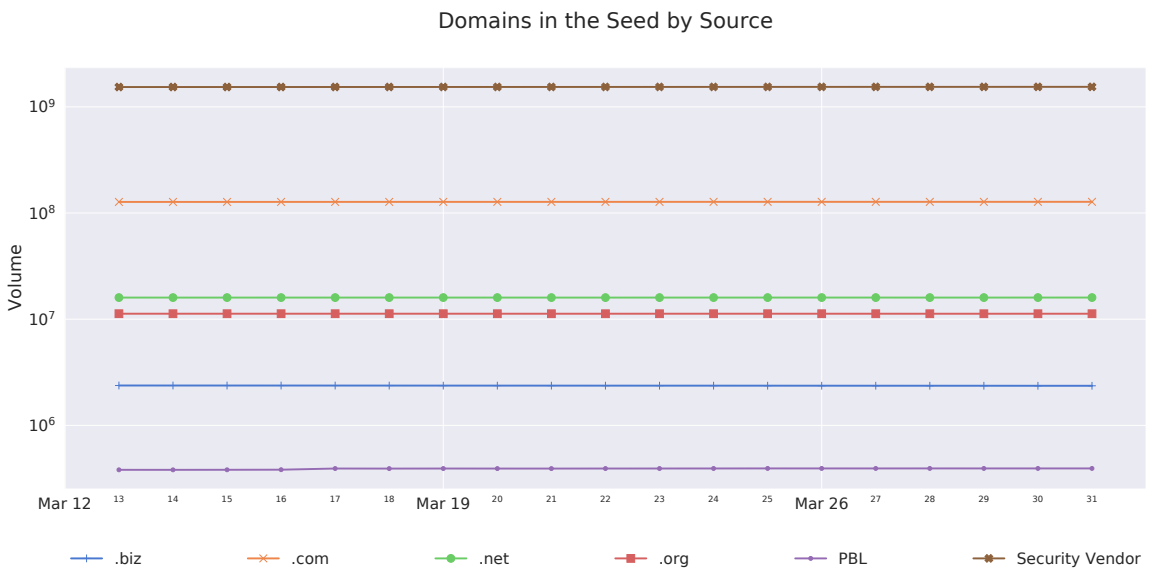


Figure 3.3: Number of domains over time per seed input. The security vendor list contains about 1.5 billion domains and from the TLDs com is obviously the largest one with about 127 million domains.

state of each zone every day.

We also add the entire Alexa [57] list of popular domains to the domain seed. This provides us with a large number of domains that would most likely be queried in a network by users.

In order to capture domains that might not be available in one of the zone files, we built a crawler that collects and parses domains seen in the Common-Crawl dataset [58]. The Common-Crawl dataset is an open repository of web crawl data that offers large volumes of crawled pages to anyone. We used components (i.e., URLs, HTML code) from the common crawl dataset to extract only the domains of the pages visited. Due to the size of even the Common-Crawl "metadata section" from the common crawl, we are still using the data published for last September 2015 and will start updating that list regularly. Because the common crawl data is published in monthly releases, the domain list that we extract from it and use in our seed list remains the same between updates.

A different list of data that we utilize in our domain seed is a feed of "interesting" do-

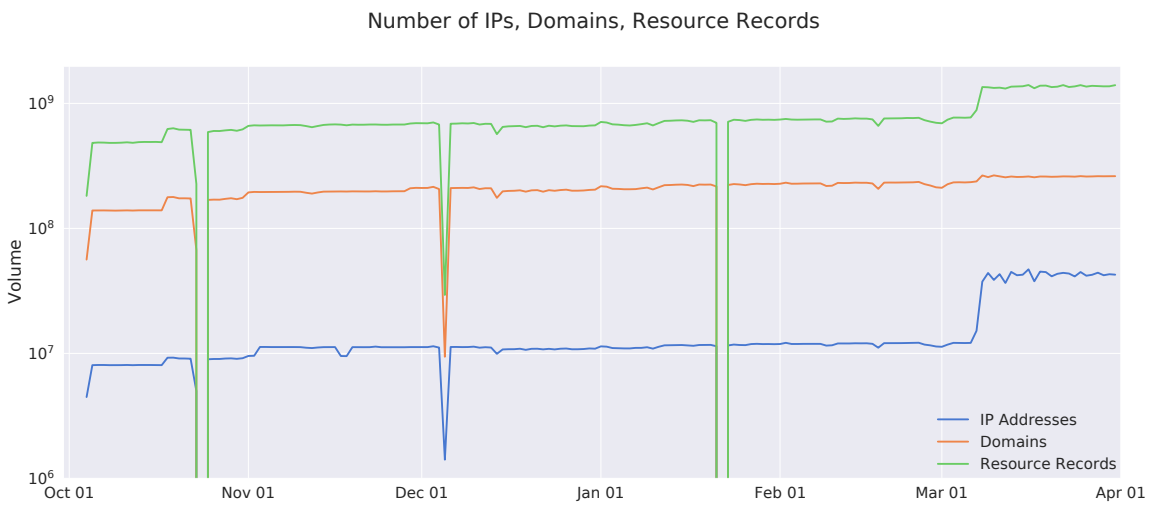


Figure 3.4: Volumes of IPs, resource records and domains observed with Thales. March 7th was the day when we started querying for the QTYPEs: SOA, AAAA, TXT and MX. There have been two full outages on October 25, 2015 and January 23, 2016. On December 6, 2015 we had an outage that lasted for most of the day but we were able to recover the system later in the day.

mains that have been provided to us by a security company. This feed provides us with domains that have been observed to engage in forms of potentially malicious Internet activity. Because the feed provides us with new domain names constantly, we gather all new information and append it to the already existing list of interesting domains. We push the updated list to our collection infrastructure daily. The feed provides us with tens of thousands of new domains each day, making this list one of the fastest growing lists we use.

Finally we use a collection of public blacklist data in order to provide our data with interesting hand curated domains that originate from malicious activity. More specifically the public blacklists we employ are: Abuse.ch [59], Malware DL [60], Blackhole DNS [61], sagadc [62], hphosts [63], SANS [64] and itmate [65]. We aggregate these lists daily and we input them into our domain seed by replacing the old list.

3.2.3 Measurements

Thales has been collecting data for a little less than six months. For the purpose of this paper we are focusing on analyzing all data in this section and then limit in depth analysis to the last 12 days of March (the last full week forth) for more specific measurements, unless a different window is explicitly stated. Over six months, Thales identified approximately 10,714,784 unique IP addresses, 199,110,841 unique domain names and 662,319,389 unique RRs per day. Figure 3.4 shows the distribution of IP addresses, domain names and RRs on average per day from October 5th to March 3rd 2016.

During these months, we experienced two outages. The first was when the system was initially setup because of an update which was not rolled out correctly and caused the system to go off-line. Therefore, there is no data available for October 25, 2015, and policy has been updated to avoid future interruption since then. On January 23, 2016, our campus data center was undergoing maintenance for the cooling infrastructure, which caused a temporary shutdown of all our systems. Such cases can now be mitigated by Thales. We have made the system portable, which gives us the ability to move it to another location within a day's prior notice. Also on December 6, 2015 early in the day we had a hardware failure on our system that was detected early in the morning. We were able to recover the system and perform a check of the system by the same afternoon. After the system check, we immediately restarted the collection, but there was not enough time in the day to go through the entirety of data in our seed list. This is depicted by the significant dip in the data. This incident was not a full outage since we were able to collect some data for the day.

3.3 Comparing Active And Passive DNS Datasets

Passive DNS has been an invaluable weapon in the community's arsenal for research combatting malware, botnets, and malicious actors [40, 17, 18, 66, 67]. Passive DNS, though,

is rare, difficult to obtain, and often comes with restrictive legal clauses (i.e., Non Disclosure Agreements). At the same time, laws and regulations against personal identifiable information (PII), the significant financial cost of the passive collection, and storage infrastructure are some of several reasons **that make passive DNS cumbersome**. The primary goal for the active DNS dataset is to reduce the barrier for (repeatable) security research on DNS. In this section, we show how active DNS relates and contrasts to passive DNS. We will see that, while not a true replacement for passive DNS, Thales is able to create active DNS datasets that in many cases contain an order of magnitude more domain names and IP addresses.

3.3.1 Datasets

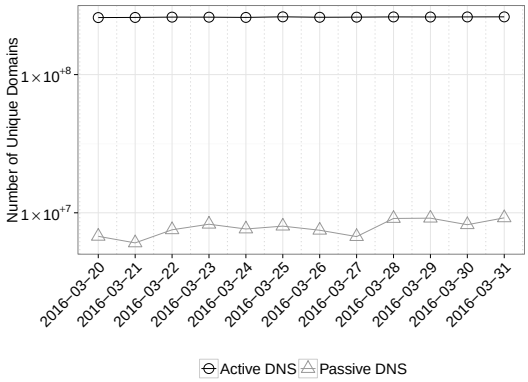
We first discuss how we obtain our passive DNS datasets. Our passive DNS dataset consists of traffic collected at our university network. The collection point is both *below* and *above the recursive*. This means that we collect the responses on the both paths; (1) between the (anonymized) clients and the local recursives and (2) between the local recursives and the upper layers of the DNS hierarchy (i.e., name servers, top level domains, etc.). For the active and passive DNS comparison, we decided to utilize datasets collected during the entire month of March 2016.

Figures 3.5 and 3.6 show eight detailed plots of the distribution of records in both our active and passive DNS datasets. Note that all plots are log-scale for the y-axis. As we can see, the active DNS dataset does not fluctuate a lot, compared to the passive DNS one. This is primarily an artifact of the collection technique, since the daily changes in the domain name seed we are using is minimal. On the other hand, the passive DNS dataset, is is primarily driven by the behavior of the users on the local network, which may fluctuate on weekends, holidays, and during certain periods such as exams. This also explains the sudden increase in traffic for passive DNS, since our campus network experienced a reduction in traffic from March 21^{st} until March 25^{th} during spring break.

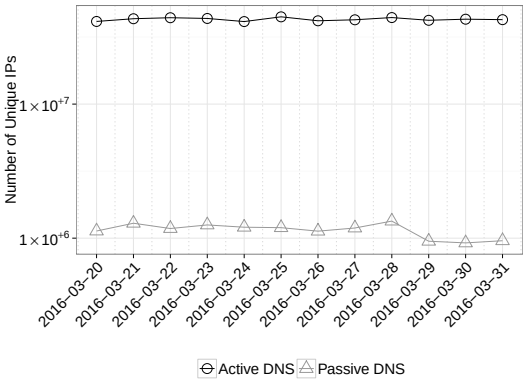
Therefore, Figure 3.5c shows an increase to more than double the unique resource records (RRs) identified per day after Monday, March 28^{th}, when the spring break ended. Table 3.1 shows a breakdown of the datasets over the last 12 days of March, in much greater detail.

It is worth noting that Thales is able to generate an order of magnitude more unique domain names, IP addresses and RDATA in the active DNS dataset (see Figure 3.5, subfigures a to e), in comparison to the passive DNS data collected in a large university. This means that in actual DNS records, the active DNS dataset is more than comparable to the passive DNS that someone can collect in a large university. Now, as we can see from Figure 3.5, (f), active DNS is not able to create as dense graphs of resource records, as someone would expect to find in passive DNS data. This is somewhat to be expected, as in active DNS, Thales is scanning all possible domain names that can be seen in our public sources. This inevitably will include domain names that are rare, and in the context of a graph compiled by RRs, they will form islands. While not necessarily bad, we would advise researchers to take cautionary sanity steps when they utilize the active DNS data for spectral processes.

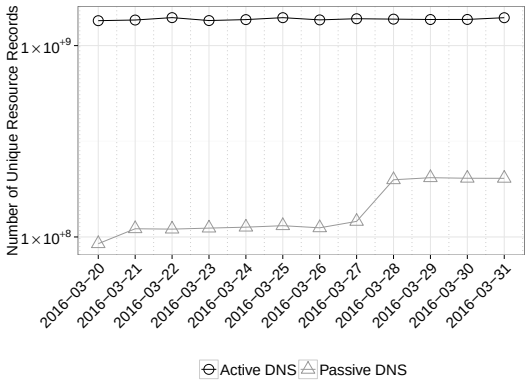
The diversity of the different query record types (QTYPEs) we are able to identify, in the two different datasets compared can been seen in the two figures in Figure 3.6. Although there is a big difference regarding the volume of the records available, on average the visibility is very similar, since we are collecting the most popular QTYPEs when querying for the active DNS datasets.



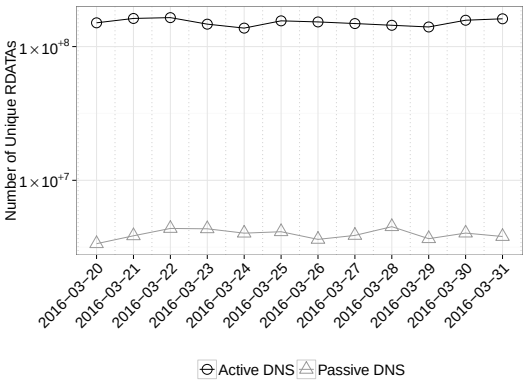
(a) Unique domain names per day.



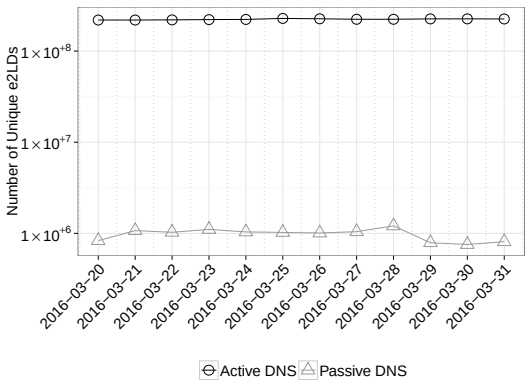
(b) Unique IP addresses per day.



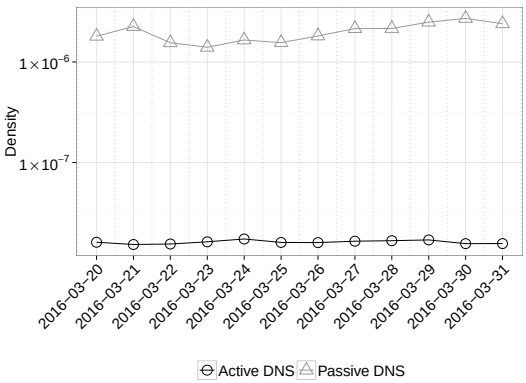
(c) Unique resource records (RR) per day.



(d) Unique responses (RDATA) per day.



(e) Unique effective second level domain names per day.



(f) The density of the Resource Records graph in the active and passive DNS dataset.

Figure 3.5: The distribution of different records in our active and passive DNS datasets. The plots show that Thales is able to generate orders of magnitude more data than the passive DNS collection engine (Figures a to e) and much more diverse (Figure f).

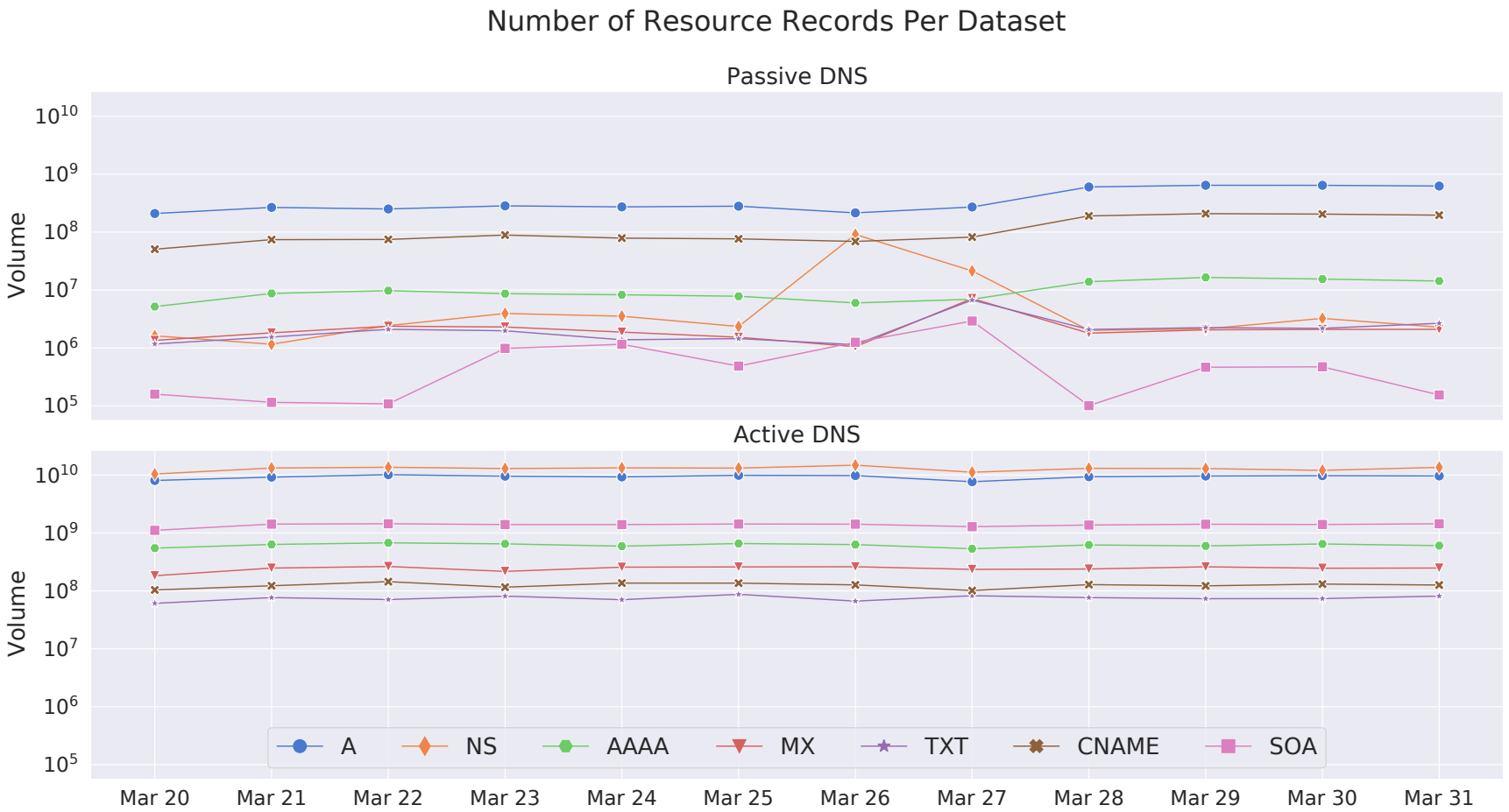


Figure 3.6: The distribution of different query types (QTYPE) in the active (bottom) and passive (top) DNS datasets. The active DNS dataset is almost sustaining the same volume of records per day, whereas the passive DNS dataset is fluctuating more over time. Note the growth after March 28, when the Spring Break was over and the Institute was operating at full capacity again.

Table 3.1: Number of data points collected over the last 12 days of March 2016. Values are in thousands ($\times 10^3$).

	Domains		IPv4/IPv6		RDATA		RR		e2LD	
Date	**Active**	**Passive**	**Active**	**Passive**	**Active**	**Passive**	**Active**	**Passive**	**Active**	**Passive**
3/20	258,702	6,759	41,360	1,130	150,629	3,356	1,350,118	92,218	219,009	831
3/21	259,305	6,056	43,333	1,292	162,366	3,845	1,360,660	110,379	219,009	1,072
3/22	260,676	7,535	44,090	1,180	164,685	4,364	1,400,427	109,896	219,985	1,028
3/23	260,420	8,267	43,538	1,255	147,190	4,338	1,352,019	111,247	221,466	1,105
3/24	259,389	7,635	41,273	1,206	137,491	4,024	1,367,554	112,513	222,464	1,037
3/25	261,883	8,008	44,769	1,197	155,830	4,125	1,399,724	114,518	228,119	1,024
3/26	260,011	7,479	41,830	1,127	152,918	3,616	1,362,978	111,646	226,030	1,009
3/27	260,506	6,727	42,556	1,190	148,728	3,871	1,382,096	120,624	223,313	1,043
3/28	261,551	9,100	44,216	1,340	144,365	4,499	1,375,399	199,023	223,345	1,208
3/29	261,171	9,145	42,189	948	140,225	3,658	1,369,100	204,017	225,513	789
3/30	261,513	8,200	42,992	921	157,477	4,030	1,370,090	202,702	225,642	754
3/31	261,766	9,195	42,651	956	161,387	3,798	1,399,218	202,511	225,128	809

Table 3.2: The distribution of QTYPEs for the active and passive DNS in our datasets.

	Aggregate ($\times 10^3$)		**Mean**		**Median**	
QTYPE	**Active**	**Passive**	**Active**	**Passive**	**Active**	**Passive**
A	3,082,960	813,485	256,913,375.92	67,790,485.33	257,181,439.5	54,989,441.0
AAAA	292,278	81,992	24,356,555.67	6,832,692.33	23,918,026.5	5,920,971.5
CNAME	174,881	136,901	14,573,484.5	11,408,450.0	14,582,732.0	8,495,216.5
MX	2,222,465	908	185,205,470.67	75,690.83	184,075,003.5	83,309.0
NS	5,822,874	586,695	485,239,507	48,891,296.25	485,117,732.0	39,316,201.5
SOA	3,498,172	28,162	291,514,366.5	2,346,885.75	291,172,940.5	2,022,850.0
TXT	701,689	14,499	58,474,102.67	1,208,253.83	58,304,209.5	1,205,094.5
Other	694,067	28,655	57,838,938.5	2,387,929.75	57,693,964	2,380,550

3.4 Case Studies

To this point, we exposed several of the data properties from the active DNS datasets. In this section, we demonstrate the security value of these new active DNS datasets. We should clarify that our goal is not to claim as a contribution any of the following abuse detection processes. All of them have been discussed by previous work in the field. Rather, our goal is to practically demonstrate, using the actual active DNS datasets, the security merit that active DNS data can offer to the research and operational communities.

3.4.1 Enhancing Public Blacklists

Due to the nature of Thales we can make use of the collected data in ways that can reveal abuse signal about domains before they are identified as actual malicious use. Blacklisted domains, for example, are an interesting category of candidate indicators of abuse that can be registered, set-up, and pointed to an IP location well before they are actually used in malicious activities. Thus, active DNS could be used as a potential source of raw datasets that can be used for timely domain abuse detection.

As we have already discussed, alongside the active DNS data collection, we were also able to gather a plethora of public domain name blacklists. As expected, domain names in these blacklists also appeared in the active DNS traces we collected using the active DNS project. For all domain names seen in both the public blacklists and active DNS data, we identified two important dates. The first denotes the first day the domain name was probed by Thales. This behavior is driven by the addition of the domain in our seed list that can be caused by a change in any of the zone files collected daily from the top level domain authorities. The second important date we identified is the first day one of the many blacklists we collect (on a daily basis) actually listed this domain name as part of a particular abusive activity.

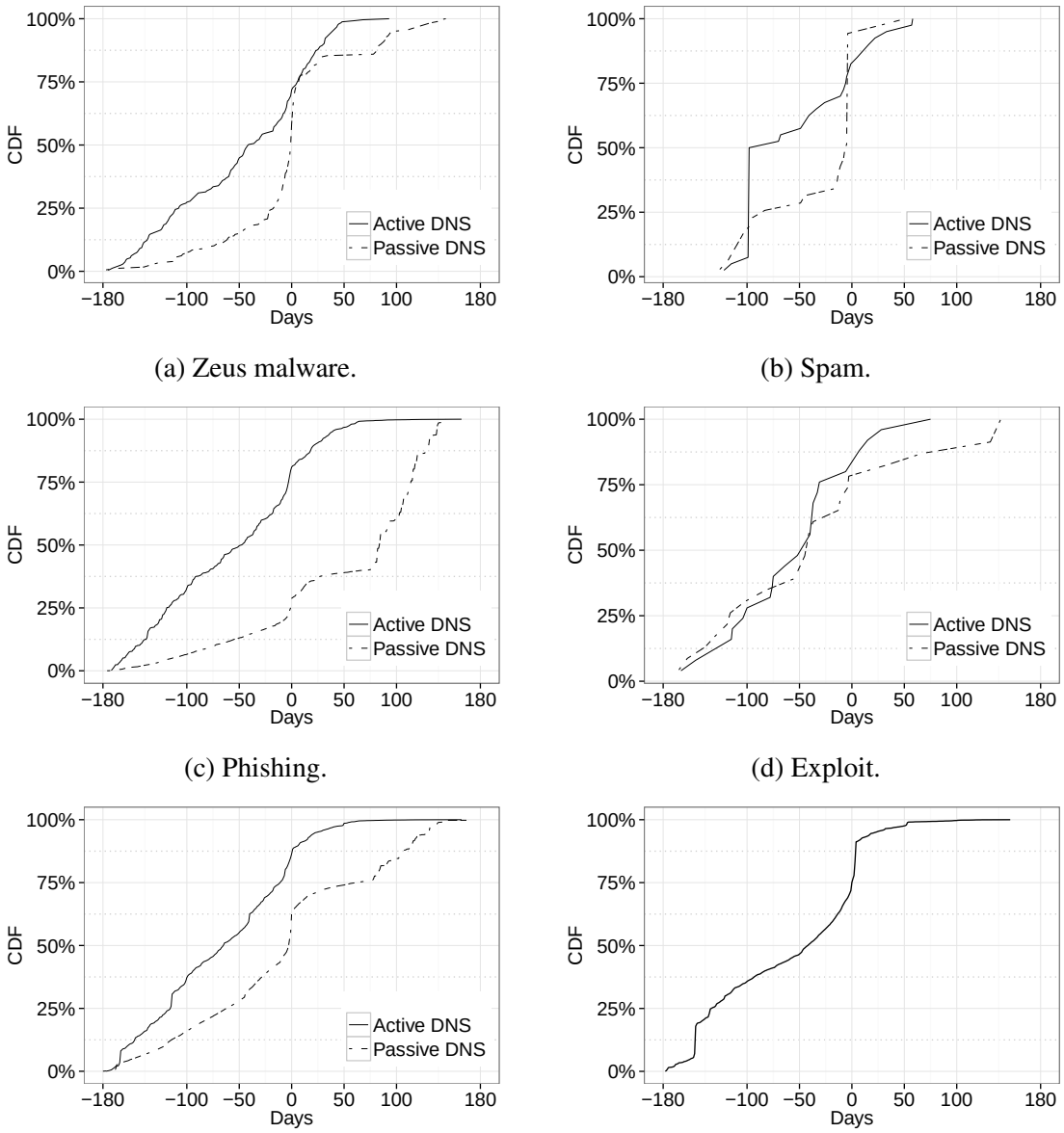
We compared the first seen dates of blacklisted domains and the first seen date of a

domain resolved by Thales and we plotted the results in a cumulative distribution function (CDF) that depicts the time difference in days between a resolution in our passive or active DNS data and the appearance of the domain in a public blacklist. Negative values represent the number of domains that have first appeared in our active or passive DNS data before getting eventually blacklisted. On the other hand, positive values represent domains that had been blacklisted before they had a resolution in our data.

It is worth pointing out that not all the public domain names blacklists were used as a seed domain source for Thales, rather the ones that are described in Section 3.2.2. That is, we should expect a fair amount of both positive and negative values in these CDFs. Positive values indicate that a domain name was first seen in a blacklist and then in either the active or passive DNS data that we present in Figure 3.7, while negative values indicate that the domain was first seen in DNS before being blacklisted.

Thales resolves domains that came in part from zonefiles for major top-level domains. It queries any domain registered in that zone within a day after it was registered and added in the zonefile. This creates a temporal history of the DNS activity capable of describing the IP infrastructure history that supported the domain name, before blacklisting, at the time, and after it was blacklisted. This is a new property that active DNS datasets will freely offer to the security community, and it is a property that is rarely seen in passive DNS data. The reason for this behaviour that active DNS exhibits compared to passive DNS is simple; infections get remediated and hosts are mobile, thus making it hard for the network operator to passively observe the network evolution of the infrastructure that supports a domain. Thus, Thales should be able to offer a strong signal augmenting existing passive DNS data to which researchers and network operators have access.

Figure 3.7 shows the CDF plots for different classes of malicious domain names (Figures 3.7a to 3.7d). The values plotted include the domains in our active and passive DNS datasets that have been blacklisted. Several instances of these domains are found in our dataset long before they are blacklisted; for example 50% of domain names associated with



(a) Zeus malware.

(b) Spam.

(c) Phishing.

(d) Exploit.

(e) Difference of days from the first time a domain name was seen in active and passive DNS before it appeared in a PBL.

(f) The difference between the first date a blacklisted domain was seen in active DNS versus the passive DNS dataset, for the domains that were seen before they were blacklisted. Approximately 70% of the 17,000 domains that exist in both datasets and were blacklisted, were first seen in active DNS.

Figure 3.7: Cumulative distribution of the first seen date in active and passive DNS, subtracting the first seen date of the same domain in a PBL for Zeus, Spam, Phishing, and Exploit domains.

spam were queried approximately 2.5 months before they were blacklisted. On the other hand, we do not have the same visibility for ephemeral types of attacks, like phishing and exploit kits. In the latter two cases, approximately 75% of the domain names are queried by Thales at least one day earlier, with the 50% mark being at around 50 days earlier.

In total 42,000 domain names have been blacklisted and also appeared in our active DNS dataset. From this set, 30% were queried and data have been collected for approximately 100 days before the blacklisting instance (Figure 3.7(e)). For 75% of the blacklisted domain names, we have collected data for more than a week before they appeared on a PBL. Considering that PBLs have been used as ground truth for various security systems [43, 68, 69, 70], we are planning to utilize this data over time to model the behavior of these domains and identify the threats long before current systems, or even before they are utilized by the adversaries.

On the other hand, we were able to identify 20,000 domain names in the passive DNS dataset that also appear in blacklists. The dashed line in Figure 3.7 plots represents these domain names. Approximately 50% of the domain names that are blacklisted appear in the passive DNS data feed, with only 25% revealing themselves 50 days earlier than the blacklisting event, as shown in Figure 3.7e. In this case, there are only 20,000 domain names that have been blacklisted and the visibility that we have is approximately 15% for the 100 days mark. About 50% of all the domain names were seen roughly two days before they were blacklisted. This clearly supports our claim about the merit of active DNS datasets, and how well they complement existing passive DNS repositories. The early linkage between domain names and IP infrastructure witnessed by the active DNS data will be able to enrich the signal that passive DNS data contains, potentially making local DNS modeling efforts easier for researchers and operators.

In most cases, the active DNS dataset contains domain names far before they appear in either the passive DNS or the blacklist dataset. Note that the intersection between active and passive DNS records that have been blacklisted is approximately 19,000. This is almost

half of the domains in the active DNS dataset and 95% of the domain names in the passive DNS dataset. Passive DNS seems to show better results in early days for the spam domain names case (Figure 3.7b), but active DNS catches up very fast (within 15 days) and then loses the advantage again at the time of the blacklisting events (0 point in the plot).

Lastly, Figure 3.7f depicts the difference between the day a blacklisted domain name was first seen in our active DNS dataset and the day it was seen in our passive DNS dataset. This includes only the domain names that were seen before the PBLs included them. Approximately 17,000 domain names have been found in both active and passive DNS before they were blacklisted. The vast majority of them were first resolved by Thales, at least one day before it was visited by a system in our university. Approximately 40% of the domain names were already being resolved by Thales for more than 100 days before they appeared in the passive DNS dataset.

3.4.2 Enhancing The Detection Of Domain's Residual Trust Change

On the Internet, domain names serve as trust anchors for numerous systems and services, and for many, ownership of a domain is enough to prove one's identity. Work by Lever et. al [71] discussed the problems caused by the use of domains as trust anchors and showed that residual trust, implicitly inherited by domains after an ownership change, is a root cause of many seemingly disparate security problems. Therefore, identifying changes in ownership, due to expiration or some other cause, is an important problem in protecting against the abuse of residual trust. WHOIS [72] is typically used to discover more information about the owner of a particular domain, and thus, it would a appear to be a natural fit for creating a remedy to this problem. However, collecting WHOIS at scale is outside the grasp of most organizations due to rate limiting imposed on automated collection of WHOIS records. To make matters worse, these limits frequently vary by registrar, further adding to the complexity of collecting WHOIS data at scale. To circumvent this problem, Lever et al., proposed Alembic, a lightweight algorithm for locating potential ownership

changes that relies solely on passive DNS. This algorithm relied upon three different components: changes in infrastructure, changes in lookup volume distribution, and change in SOA records.

While passive DNS is much easier collect, it is also very sparse, and this results in two limitations with respect to Alembic. Scores can only be computed for domains observed in passive DNS and that have sufficient historical resolutions. Active DNS can help improve upon these limitations. First, Figure 3.5e shows that active DNS captures many more effective second level domains than passive DNS. Given that the passive DNS dataset used for comparison was generated from a large university network, this result is particularly important. It demonstrates that even large networks have difficulty matching the breadth of domains that can be collected using active DNS querying. Next, active DNS querying can consistently gather specified DNS record types over time. In particular, the two plots in Figure 3.6 show that active DNS results in substantially more SOA records than passive DNS each day. Since one of the key components of the Alembic scoring is SOA records active DNS should be able to enhance the performance of the Alembic scoring algorithm. While active DNS provides many benefits, it is important to note that the one component Active DNS cannot enhance is the lookup volume distribution of domains. This component is derived by user behavior observed in passive DNS, and therefore, there is no analog in the active DNS dataset.

To evaluate whether Alembic could work using only active DNS, we implemented a modified version of the algorithm that excluded lookup volume distribution as a component and used a fixed window size of two weeks. Then we computed scores for March 27, 2016 using our modified algorithm. In total, this resulted in 63,332,836 domains with non-zero scores, where larger scores indicate a higher confidence in an ownership change. The distribution of those scores can be seen in Figure 3.8. The majority fell in the range between 0.4 and 0.5, and further inspection revealed that the SOA component contributed the most to these scores. In short, most of the scores in this range were a result of changes

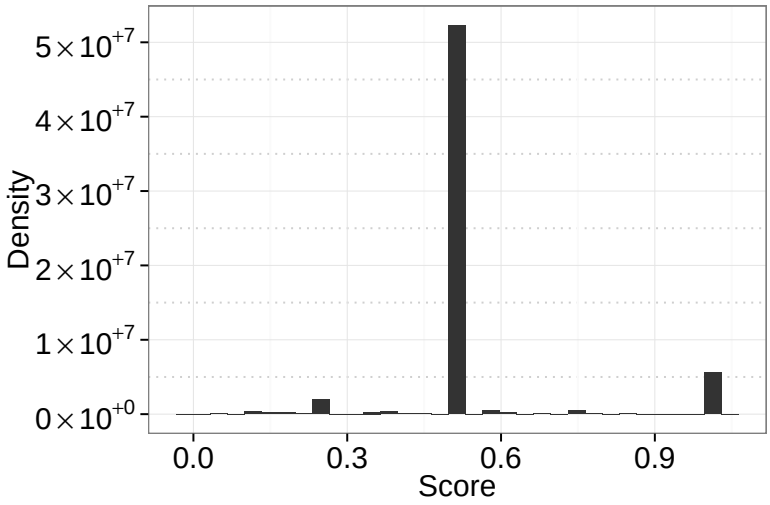


Figure 3.8: Histogram showing the distribution of Alembic scores for March 27, 2016.

in the SOA record for the domains. Since we saw very little change in hosting infrastructure, it is possible these scores could simply be the result of minor changes within the SOA record. The next largest range was between 0.9 and 1.0 and consisted of 5,652,910 domains. According to the algorithm, domains with a score in this range are most likely to have undergone a change in ownership. 5,625,397 (99.5%) of these domains had a score of 1.0, indicating that both infrastructure and SOA records had undergone complete changes. Indeed, we found 10,885 of these domains on a public service's list [73] of expired domains for March 27, 2016. The remainder of these domains provide interesting cases for further study.

Our modified version of the Alembic algorithm, originally proposed by Lever et al., provides an interesting example of how active DNS can be used to enhance or extend existing research. Without active DNS, deploying an algorithm like Alembic would require access to a large scale passive DNS dataset (e.g., university, enterprise, Internet service provider). However, using openly available active DNS data, as offered by this research, can help remove the barriers to using or deploying existing DNS research.

3.4.3 Tracking Malicious Domain Names In Non-routable IP Space

Bogons are private, reserved, or otherwise unallocated network blocks [74, 75, 76]. Bogons should be boring since by definition they should not be hosting anything in the context of the global Internet. But occasionally, a domain name, like `messisux.bix`, resolves to a bogon like `0.0.0.0` despite the fact this IP can not host anything. The presence of a domain name, however, indicates a service that should be globally reachable exists. These "nonsense" resolutions are at times caused by misconfigurations, brand protection services, and occasionally, malicious actors. To investigate further, we don our threat researcher hats and analyze domain names that resolved to bogon IP space during our analysis. Here we focus on malicious infrastructure as it is a primary interest of the security community. However, we also note that active DNS data that resolves to bogons would be useful in other contexts such as identifying potential trademark infringements.

We identified two known malicious campaigns in the subset of bogon data: "Operation Hangover" and "CopyKittens." The former is infrastructure of a cyber espionage threat targeting government, military, and private sector networks with some ties to India [77]. Domain names seen in active DNS data for this threat are shown at the top portion of Table 3.3. The latter is infrastructure for threats targeting "high ranking diplomats at Israel's Ministry of Foreign Affairs and some well-known Israeli academic researchers specializing in Middle East Studies" [78] and its active DNS domains are shown at the bottom portion of Table 3.3.

These are useful indicators despite the fact these attacks are known and likely inactive. Neutered, yet unidentified, infections are likely still operating in networks today, which should lead to incidence responses and damage assessments. For example, knowing the specific internal machine that was infected with targeted malware is useful even after an attack has taken place. An end-user machine on a company's corporate network has different implications than a locked down server in a data center, or the CEO's personal laptop. Interestingly, some targeted threats do resolve to bogon space, while active, to reduce their

Table 3.3: Operation Hangover and CopyKittens Attack Group Infrastructure and Domain Names.

Operation Hangover
alertmymailsnotify[.]com
cloudone-opsource[.]com
download-mgrwin[.]com
necessaries-documentation[.]com
newsfairprocessing[.]com
onestop-shops[.]com
servicesloginmail-process[.]com
servicesprocessing[.]com
websourceing[.]com
worldvoicetrip[.]com

CopyKittens
alhadath[.]mobi
big-windowss[.]com
cacheupdate14[.]com
fbstatic-akamaihd[.]com
fbstatic-a[.]space
fbstatic-a[.]xyz
gmailtagmanager[.]com
haaretz[.]link
haaretz-news[.]com
mswordupdate15[.]com
mswordupdate16[.]com
mswordupdate17[.]com
patch7-windows[.]com
patch8-windows[.]com
patchthiswindows[.]com
walla[.]link
wethearservice[.]com
wheatherserviceapi[.]info
windowkernel[.]com
windows-drive20[.]com
windowskernel14[.]com
windows-my50[.]com
windowsupup[.]com

network footprint [79]. This suggests signal for malicious detection in active DNS's non-routable IPs.

3.5 Conclusion

DNS is vital to the operation of the Internet. Users, systems, and services rely on its operation for most network communication—often without even realizing it. Malware is no different. It makes use of DNS to locate C&C servers and provide network agility. Despite all its uses, it is incredibly difficult to gain access to large, open, and freely available DNS datasets, and even when possible, such data is often encumbered with privacy regulations or access restrictions. This severely limits the pool of security researchers than can leverage DNS in their work. Furthermore, it limits the repeatability of existing DNS based research. Clearly, there is a need in the research community for access to large, open, and freely available DNS data. To that end, this work built a new system, Thales, to query and collect massive quantities of DNS data starting from publicly available lists of domains (e.g., zone files, Alexa, Common Crawl, etc.). We are releasing the resulting active DNS data from this system to the public, and since this data is derived from public sources, it can be easily incorporated into new or existing research without having to worry about privacy regulations or access restrictions.

To prove its merit, we provide an in-depth comparison between active DNS and a passive DNS dataset collected on a large university network. This analysis showed that active DNS data provides a greater breadth of coverage (i.e., greater quantity and greater variety of records), but passive DNS data provides a denser, more tightly connected graph. Due to these differences, we provided case studies demonstrating how active DNS can be used to facilitate new research or even re-implement existing DNS related research. It is our sincere hope that by opening up active DNS to the security community we can spur more and better research around DNS.

CHAPTER 4

ACTIVE DNS: THE FIRST QUINQUENNIUM

4.1 Introduction

The Active DNS [1] data, since its first public release in 2015, has become an important tool in researchers' toolkits. More than 70 different entities across academia and the private industry obtain the data and use it in network and security research [27, 80, 81, 82, 83, 84]. The work from Kountouras et al. [85], in 2016, presented the value that the Active DNS data brings to the community and described volumetric characteristics and case studies where the data could provide significant value.

However, after operating the Active DNS data collection system, Thales, for almost five years, we have experienced several issues and problems that were being addressed as they manifested themselves. Most of these issues revolved around the robustness and availability of the system and not the resulting dataset. Unfortunately, when the system's integrity was compromised, for instance during a network outage, then the data integrity was at stake as well. Hence, we have made several architectural changes to the original system, depicted in Figure 3.1, p. 28, which we expect to provide much better performance and much more data coverage, while minimizing user involvement and operational oversight.

This chapter provides a detailed description of the issues we have faced over the years and the way they have been dealt with. It explains the architectural changes we have made and outlines the new architecture for the new Active DNS data collection system, Thales 2.0. As we will show later, the disadvantages of Thales, which led us to the need for an upgrade, revolve primarily around infrastructure and data management. We should note that architectural decision made in the design and implementation of Thales were bounded

[1] https://www.activednsproject.org

by the technology available at the time and resource constraints imposed by the environment within which the system ran. Several components, like the DNS querying engine for example, had to be overhauled in the process. However, we thoroughly evaluated changes to facilitate a new system that can produce at least the same quality and quantity of data.

The changes to transition from Thales to Thales 2.0 took place over the course of approximately a year and we have been running both systems for approximately a year in parallel to quantitatively and qualitatively verify the data collected by Thales 2.0. Sections 4.5.1 and 4.5.2 present the respective quantity- and quality-based analysis and demonstrate the value added to the data both in terms of raw DNS records, and in terms of significance for security research, similar to Section 3.4, p. 40.

We expect that the transition to the new system, Thales 2.0, will increase the availability and reliability of the data, as well as the sharing capabilities and applicability to research. This chapter makes the following contributions:

- We describe issues that we came across while operating Thales for almost five years. We go over the technologies that were used and how we changed core components of the system to tackle problems with orchestration, scalability, and deployment.

- We demonstrate how newer available distributed computing technologies have enabled Thales 2.0 to horizontally scale much better than Thales and integrate with a distributed Extract, Transform, Load (ETL) pipeline. New technologies allow for further collaboration and system deployment to remote locations, enabling potential *looking glass* capabilities for DNS data.

- Finally, we demonstrate the quantitative and qualitative value added by Thales 2.0 to the Active DNS data. We show that the new Active DNS data has an order of magnitude more records than before, and that it can provide data points days, or even months, before Thales, which can be a significant advantage in security research and applications.

4.2 Thales 2.0 Architecture

This section describes the architecture of *Thales 2.0*, the *current* version of the Active DNS data collection and processing system. Over the last few years, Thales has evolved and significantly changed to accommodate requirements either from our team or from external sources, given the public nature of the resulting datasets.

The initial Active DNS data collection system (Thales) was designed in 2014 with technologies available at the time and requirements set by our research team. However, over the years, many new technologies have become available and we identified cases where we can better the system. The overall architecture has not changed significantly. Figure 3.1 depicts the architecture of Thales. For comparison, Figure 4.1 represents the architecture of Thales 2.0, where we can see the *seed generation*, *data collection*, and *permanent store* from left to right.

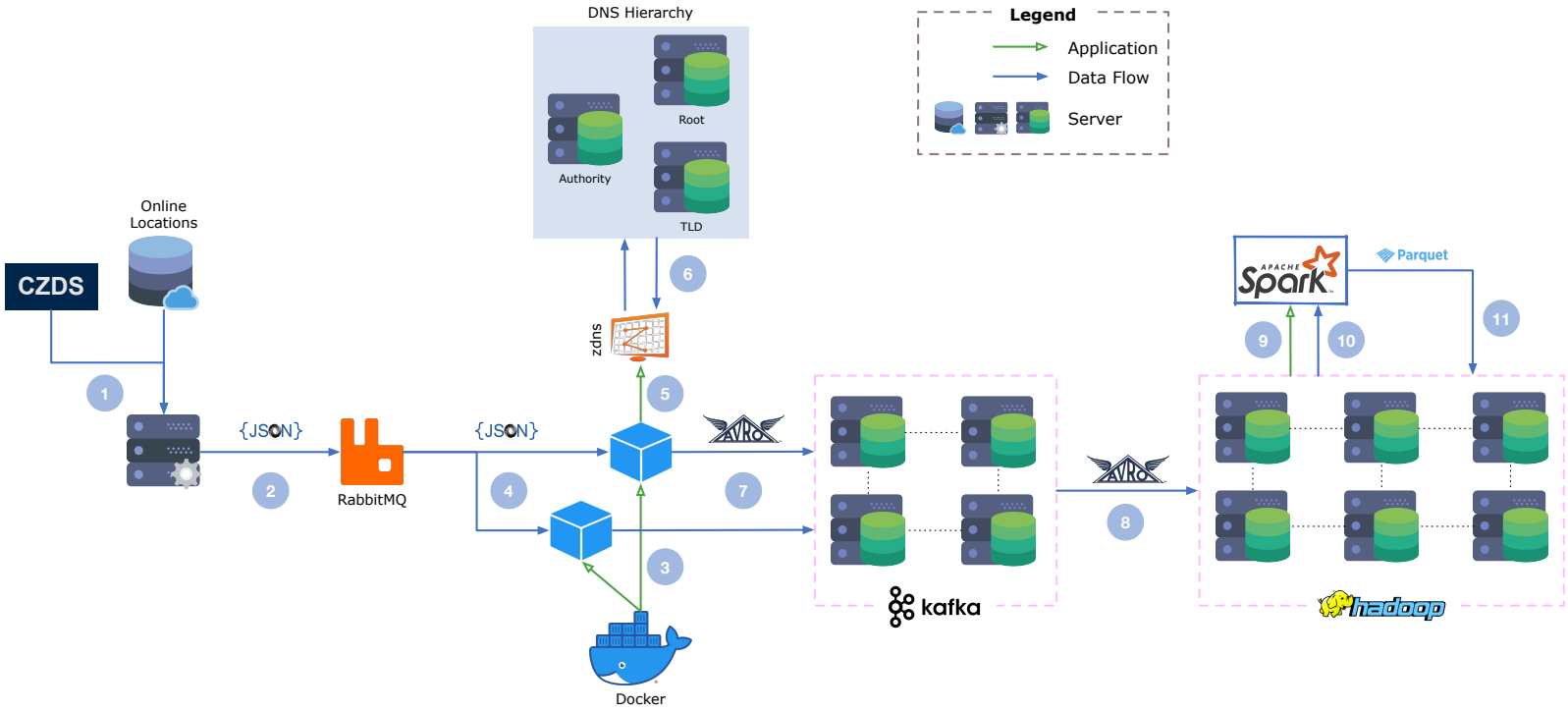


Figure 4.1: Overview of the *current* Active DNS system architecture, Thales 2.0.

At the very first step, 1, the system generates the seed of domain names that will be resolved. Like previously, we obtain popular Zone Files from large TLDs (e.g., `com`, `org`, `net`, etc.) and data provided to us by security companies. We have extended our list of domain names in the seed by subscribing to the Centralized Zone Data Service (CZDS) [86], an online portal where interested parties can request access to Zone Files from participating generic-TLDs (gTLDs). This provides us access to more than 1,000 new TLDs where we can obtain every registered domain they are responsible for.

Right after collecting the domain names that will be queried, we organize them and push them to a RabbitMQ [87] First-In, First-Out (FIFO) queue at step 2. The RabbitMQ message queue is a central component of the new system that is responsible for managing domains to be looked up (i.e. the seed list). RabbitMQ is a very powerful messaging platform that supports Access Control Lists (ACLs) for multiple users, asynchronous messages, and FIFO queuing out-of-the-box. RabbitMQ has a very wide user base, is actively maintained, and easy to deploy and manage. RabbitMQ has client implementations in many popular languages, including Python, Golang, Java, C, Ruby, and others. The wide selection of client libraries combined with the powerful features of RabbitMQ made it an obvious choice given our use case. A client application handles submission of domains to RabbitMQ. It is written in Golang [2], which allows us to generate a small binary that is easily distributed, either by itself, or in a lightweight Docker container, so that third parties or third party applications can contribute domains to the RabbitMQ queue.

The *worker* is at the core of the contribution to ActiveDNS. This is the component of the new system that is responsible for performing the DNS lookups. The worker is implemented in Golang. Golang was chosen because of the powerful DNS libraries that have been implemented in it, as well as for the performance of the language itself. The worker makes extensive use of Miek Gieben's DNS library [88] and ZDNS [89] to perform and process resolution requests. Workers are *containerized* for simple deployment through

[2] https://golang.org

Docker [90]. Containerized applications are simple "contained applications" which operate atomically to perform some functionality and are then discarded. This allows us to deploy as many applications as we need in order to maximize our data collection rate, while minimizing the maintenance "cost".

A collection of commodity hardware servers are using *Docker Engine* on a Debian-based Linux distribution to instantiate workers. Hence, at step 3, we expect the aforementioned servers to startup containerized workers, which will take care of resolving domain names. The worker will first pull a batch of domains, at step 4 from RabbitMQ. It will then utilize ZDNS 5 to perform the DNS lookups iteratively and process the incoming DNS packets from the DNS hierarchy 6. This processing involves extracting the relevant information from the DNS packet, namely the Resource Records (RRs), which are then reshaped and restructured along with their respective metadata. The resulting records are formatted into Apache Avro [91], a data serialization system that allows for structured and atomically contained records. Avro makes the next step much more efficient and is a Hadoop native format that is particularly important for our long-term storage. Finally, the worker will push these Avro records to our Confluent Kafka [92] cluster for temporary storage 7.

Kafka is a powerful log aggregation and data streaming platform. One of the hallmarks of Kafka is its performance and scalability on commodity hardware. We knew we would have many terabytes of data to process, based on volumes of data generated by the original system. Kafka has been shown to easily handle volumes of messages well in excess of what we generate on relatively small clusters [93, 94, 95, 96]. Kafka has an extensive user community and Confluent maintains its own distribution that is actively supported. We selected Confluent's distribution of Kafka over the Apache one because of the tools that ship with it. Confluent includes a tool called *Kafka Connect*, which provides a seamless way to interface between the *Hadoop Filesystem* (HDFS) and Kafka. In addition to Kafka Connect, we take advantage of *Confluent Schema Registry*, a service that allows for cen-

tralized management of data schemas and transparent deployment across different services (e.g., Hadoop, Kafka, Spark, etc.). This makes moving the ActiveDNS data from Kafka to HDFS robust and fault tolerant. The data is pushed to HDFS at step 8, once the Hadoop cluster is ready to ingest it.

Finally, once the Apache Avro data has been copied from Kafka to HDFS we use Apache Spark [97] to convert the Avro into Apache Parquet [98], a columnar storage format (in contrast to the row-oriented Avro format). For the Active DNS data, Parquet is a farm more efficient way to store data, since it can achieve very high compression and reduction rate, when there is a small number of columns for a very large number of rows, especially when values in each column are repeated. As we will see later in Figure 4.2a, p. 71, there are several boolean and string values that are expected to repeat themselves throughout every individual partition of data. Therefore, in step 9, the Hadoop cluster will instantiate Spark workers that will read the Avro data, generated by Thales 2.0 10, from HDFS and write the resulting output back to HDFS 11 for long-term storage.

The redesigned architecture provides several benefits to the Active DNS data collection, some of which solve significant problems discussed in Section 4.4, but most importantly, allows for increased availability and extensibility. Therefore, at the time of writing, it is particularly easy for Thales 2.0 to be extended to query more and more domain names as we see fit, we can add new QTYPEs we might consider interesting in the future or per request, and we can allow third parties and collaborators to simply submit domain names to be queried to our global queue. Hence, we manage to open the system even further to the security community, to not only use the data that we produce, but also contribute to the data that the rest of the community will be able to use.

4.3 Challenges With Thales

The new architecture of the Active DNS data collection system has provided significant benefits and has solved various issues that were uncovered over the years. This sections

dives deeper into the different problems we faced and how they have been resolved with the architectural decisions we made for Thales 2.0.

4.3.1 Data Collection & Temporary Storage

In the original system, data was collected via a *network span port*. As we can see in Figure 3.1, the Network Span was expected to mirror traffic destined to the LXC Farm (a), to the Collection Point (b). This replication, allows us to simply run a network packet capture tool (e.g., `tcpdump`, `pcapdump`, etc.) and collect data in `PCAP` format. The PCAP format is very popular for network data dumps and easy to use for network traffic analysis. However, it starts becoming increasingly difficult to use as the data size grows and processing needs to be almost as fast as the collection.

Our original *Extract Transform Load* (ETL) pipeline pushed the aforementioned PCAP files from the collection point to HDFS, where they were temporarily stored. For backup purposes, we kept a rolling window of the last 14 days of data in PCAP format. Given the sheer size of the Active DNS data (approximately more than 2TB of network data per day), we were using more than 80TB of disk space (accounting for a 3x replication on HDFS for availability and disaster recovery) at any given point.

Once the PCAP files were in HDFS we needed to convert them to a more Hadoop friendly and user-oriented format for researchers to access it. Therefore, we needed to *parse* the data into a format like Avro [91], which would allow for faster and easier use on HDFS. At this point, we should note that Hadoop shines when data can be split into small units that can be processed individually and independently, and then merged again, like in the *Map-Reduce* (MR) paradigm. Unfortunately, PCAP is an unsplittable file format, which means that every individual file needs to be processed by a single worker. One can immediately imagine a significant bottleneck. If parsing 1GB of data in a PCAP takes 10 minutes, then processing 300 files of 1GB each (300GB in total), in parallel (optimal for Hadoop), would take 10 minutes. However, if we were to process 200 files of 1GB each and a single file

that is 100GB (still a total of 300GB), the process would now take 1,000 minutes, or more than 16 hours.

In our case, the distribution of file sizes was not skewed towards a particular file, since Thales used to be working for almost all 24 hours of a day. However, our parsing performance was bounded to the number of PCAP files we had, rather than the number of available processing units on our Hadoop cluster; the later was always multiple of the former. For our parsing needs, we utilized texttttthadoop-pcap [99], a PCAP *Serializer-Deserializer* (SerDe) for Hadoop, built and maintained by RIPE-NCC. Unfortunately, the last time this library was actively being maintained was in 2016 (more than a year *after* the original Active DNS system), which made us reconsider it being in our critical path. We had two options; either maintain a fork of the library ourselves, or switch to a different approach. Since the library was not Hadoop native and never picked up by Apache itself, and given the deficiencies discussed earlier, we chose to not use this SerDe any more, neither waste computing power on our Hadoop cluster for PCAP parsing. Instead, we initially built a Nomad [100] cluster specifically for parsing PCAPs so we could move this entirely out of Hadoop. In fact, the use of the containerized microservices that submit the DNS queries (Figure 4.1, 3), allowed us to completely remove PCAPs from Thales 2.0, that provided us with even more benefits, discussed later.

4.3.2 Data Size & Data Transfer

The original Active DNS data collection system, as mentioned earlier, used to be collecting data using the network PCAP format. Since PCAP contains data from every network stack layer, the data files tended to be drastically large. Given a network packet, a lot of data points would go unused for the purposes of the Active DNS data itself, that was later made available to the community. For example, the Open Systems Interconnection (OSI) model Layer 1 and Layer 2 information, the destination IP address of the packet (our servers), most of the Layer 3 headers, are some of the data points that are being discarded after

parsing the PCAP format into what Active DNS stores long-term. The total size of the aggregate PCAPs for a day's worth of raw Active DNS data is approximately 2TB.

The data collected at the Collection Point (Figure 3.1, (b)) was transferred to our Hadoop cluster. Data transfer between clients and HDFS in the Hadoop ecosystem is not optimized by default and could take a considerable amount of time, depending on the data size. In our case, we had to wait for several hours until 2TB of data was transferred between the Collection Point and HDFS. Several solutions to that problem have been built over time, including Flume [101], NiFi [102], Gobblin [103], Kafka [92]. Most of them, however, require structured data and do not perform well with binary data files they cannot process (e.g., PCAP). Hence, moving away from PCAP files also provides a significant benefit towards using tools that are made particularly for use cases like ours.

Finally, the large PCAP files, introduce another burden to Hadoop. Since HDFS will need to replicate files (at least three times in our case), transferring large files to HDFS, makes nodes that store these files to also transfer them to at least two other nodes. Large files will take longer to transfer between HDFS data nodes, which might result in timeouts, or availability issues, that will force a new transfer to another node, over-utilizing the network and spending resources in vain. Even if the files were smaller, we would still be attempting to transfer 2TB to HDFS nodes, which would in turn transfer four more terabytes to other nodes. Considering that the Active DNS resulting dataset is only around 100GB per day, we are looking at an overhead of 5.9TB transferred and processed on Hadoop.

These problems are solved in the new architecture because of the preprocessing happening before we send any record to HDFS. Since the new querying engine can reduce data to individual records, we reformat the data into Avro [91], which is both splittable and smaller, and then write it to HDFS, reducing the data exchanged to a non-negligible 2.5TB down from 6TB.

4.3.3 Orchestration

An important aspect of the Active DNS system is the seamless and transparent operation. Our initial system, Thales, had the very basic principles of distributed systems, in an attempt to abstract as much of the day-to-day operation from the users. However, there were significant steps that could be taken towards fully automating the system and making it not only continuously available, but also fault tolerant and disaster recoverable.

LXC Container Management

When development began on the original system, container software was still in its very early stages. The first version of *Linux Containers* (LXC), namely `LXC 1.0.0` was released in February 2014. At the same time, Docker, another container management engine, was using LXC to support containers and containerized applications. Our decision to use LXC over Docker at the time, was primarily driven by the lack of Virtual Local Area Network (VLAN) support in the Docker ecosystem. LXC was closer to the "bare metal" implementation, which had benefits, like the support for VLAN mapping to containers, but also drawbacks, like the very limited command line tooling and orchestration.

Since VLANs were necessary in the implementation of Thales, as noted in Section 3.2.1, we manually built a toolkit to overcome the LXC limitations. Unfortunately, as Thales grew, so did the requirements we had from an operational standpoint (e.g., scaling, uptime, etc.), making the Active DNS project large enough to require much more attention on a daily basis, than the capacity of our research team. For example, unexpected power outages at our datacenters required manual effort to restart services and containers. Similarly, fallback mechanisms for LXC failures were not put in place by LXC itself, which mandated more and more management scripts from our side for contingency. This process resulted in a lot of ad-hock code, written to patch issues in the underlying software itself. Maintaining that code was a significant burden that had to be alleviated.

Seed Management

At the very beginning, the seed for Active DNS was pretty small and only consisted from a handful of Zonefiles that we were downloading on a daily basis. As the program grew, more private parties wanted to provide domain names and more registrars allowed us access to their Zonefiles. Therefore, our initial plan, that consisted of a script to download, process, and format Zonefiles into a seed-list for Thales, started to also get out of hand. Every different source of domains would have a different way to make data available to us (either we had to pull data or they had to push data) and the final seed list started growing significantly. Eventually, splitting a list of domains into the number of available *worker nodes* was not enough.

Workers would unexpectedly "die", or a server with several workers would stop operating, or network connectivity would become unavailable, were just some of the problems any distributed system can face. Thales was effectively no different, therefore, such issues were devastating for recovery and made managing the seed particularly cumbersome. Thus, even though the initial design was expected to support a relatively small and stable number of different seed sources, we quickly realized that it would be very hard to maintain this process long-term and with more entities committing data.

Resolver

`Unbound` [104] was the recursive we used to facilitate the actual DNS lookups in Thales. This is a tried-and-true solution that certainly works, but it required us to separate the task of submitting a DNS resolution request from the lookup mechanism itself. That means there were two distinct services running in a single container, going against the containerization philosophy and introducing two points of failure into every container. Having multiple possible points of failure in a container can defeat the purpose and make troubleshooting much more difficult. A potential failure scenario given this setup would be if the Unbound process dies inside the container, but the code going through the seed and submitting requests

continues for long before realizing Unbound is not working. In that case we would lose data and have no way of recovering lost ground. This setup was also the primary reason for ingesting data in PCAP format during the first phase of our ETL pipeline.

As discussed in Section 3.2.1, we were submitting DNS resolution requests asynchronously to Unbound, in order to achieve maximum capacity and for speed benefits. In short, our application would asynchronously submit a DNS resolution request to the local Unbound process running on the same container and never wait for the response to arrive. Therefore, we were simply "blasting" a recursive with questions, responses to which were simply ignored. Section 3.2.1 details how the collection of the DNS responses was taking place, via network traffic collection software. Another alternative was to turn on Unbound logging of DNS queries and responses, but that seemed to degrade performance beyond acceptable levels in order to perform lookups across our entire seed.

4.3.4 Data Collection

One more important part where Thales was lacking, revolved around the *Transform* and *Load* parts of our ETL pipeline. The low-level architecture and lack of proper tooling when the system was initially built, led to problems with data integrity, availability, and sharing.

Integrity & Availability

Collecting data as packet captures meant we needed a vantage point with visibility into the entire VLAN space we used for DNS lookups. Naturally this meant we had to further decouple the software performing lookups from the services responsible for collecting their results. If the server collecting data across the span port (Figure 3.1) went offline there was no straightforward way to signal the LXC containers performing lookups to pause until the collection was properly handled again. During that window we would lose all data coming in, namely DNS responses that carry the valuable data, and our seed would also continue to be exhausted. Adding redundancy to this part of the collection phase would have been

expensive and linearly scaled the amount of data we would need to transfer and process on our Hadoop cluster.

This collection scheme also meant we had no way to validate an one-to-one mapping between queries and responses. Given that queries were simply handed off to the Unbound recursive and the responses were never inspected for completeness, there might have been query responses that we have not recorded, or there might have been query responses that we never asked a question for, like *Internet Background Radiation* [105, 106]. This last part, effectively opens the system to potential cache poisoning, since we just collected any DNS packet that crossed our span port without validating whether or not a given DNS response had been solicited. Finally, we would also miss instances where an authority fails (e.g., `SERVFAIL`, `REFUSED`), but could respond at a later time had the query been performed again after some backoff period.

Schema

The schema previously used by Thales (Figure 3.2, p. 30), was designed based on anticipated use cases for the Active DNS datasets and considering our limitations given the query generation and response collection. Over the years, we came across several different requirements either from our own research or based on what data consumers needed.

While some fields stored were intuitive, such as `qname` and `rdata`, others were more convoluted. For example, the `hours` field is a 24-bit encoding of the hours of the day during which a domain was looked up. Simply put, every bit represented one hour bucket in the 24-hour clock, which was set to one if the query response was received within that hour and zero otherwise. This was both expensive to compute and cumbersome to work with at scale, especially in big data settings.

After realizing the complexity of this single field we put example code on our website (`https://activednsproject.org/about.html`) for collaborators to work with it. Feedback that we received indicated that few entities ever used this field and found little

value. Similarly, the `authority_ips` field was fairly complex as well. It denoted the authorities that replied with that particular RR, in order for the user to be able to identify mismatches in expected responses and observed data (e.g., cases of cache poisoning). However, the lack of more context around the RR, made this field less useful than initially thought it would be. For example, such context would have been provided by including the section of the packet a given RR was found in (i.e., `AUTHORITY`, `ADDITIONAL`, or `ANSWER`). Today, this field has been moved to an `ip_src` field, which indicates where that specific record came from. Using the source IP as is, removes an aggregation step that was required in order to compute the `authority_ips` field for every tuple of `qtype`, `qname`, `rdata` in the dataset.

The schema was initially designed for simplicity and with a small amount of information, which, unfortunately, made it incompatible with other schemas commonly used for passive DNS due to the aforementioned fields. One of the goals of the project is to replace passive DNS data sources, and having an inconsistent schema makes that integration less seamless, so the redesigned schema was intended to help bridge that gap.

Long-Term Storage

Once the data had been collected and processed we stored it long-term in Apache Avro [91] format. Avro is Hadoop native and widely supported among libraries built on top of Hadoop, so it was a natural choice given our technology stack. Avro was available for HDFS when Thales was initially designed. Parquet, the format used at the moment, was at very early stage and had its first stable releases around the same time when Active DNS was becoming stable.

Avro is a row-based format, which makes it very easy and fast to write into, but very expensive and slow to read from. Therefore, it is a very commonly used format for data exchange, since it will provide a certain level of data summarization and compression very fast, but is not recommended for long-term storage any more. That is primarily because

of Parquet, a columnar storage format, which achieves much faster data access and data compression, at the cost of slower and more expensive data writes.

4.3.5 Altera Pars

The aforementioned issues are significant drawbacks to the original Active DNS data collection system. In fact, we have made radical architectural changes based on these issues. However, these do not diminish the value in the Active DNS data itself. We had been collecting a massive amount of DNS data, addressing integrity and availability issues manually, making sure that the data collected will be adequate for security research. As we demonstrate in Chapter 3, the Active DNS data collected in the first six months the system ran for, was not only a viable alternative for passive DNS, but also an invaluable source of intelligence when it comes to cyberthreats.

However, since human interaction with a highly distributed system should be minimized in order to achieve higher reliability, we performed the changes mentioned in Section 4.2. The following section describes the benefits we get from the updated architecture.

4.4 Redesign

Each part of the new architecture in Thales 2.0 (Figure 4.1) was chosen carefully to address the issues mentioned above in Section 4.3. In this section, we detail how the seed generation, the orchestration, data processing, storage, and schema have been updated to either increase availability and overall reliability of the system, or accommodate research needs. Table 4.1 summarizes the technologies used in the components between Thales and Thales 2.0.

4.4.1 Seed Management

The first issue that had to be addressed revolved around the seed management. In the original system static files were split among the LXC containers and read linearly by container

Table 4.1: Issues and related components from Thales and Thales 2.0.

Issue	Components	
	Thales	**Thales 2.0**
Seed Mgmt	Split & Ship	RabbitMQ
Orchestration	LXC	Docker w/ Docker Compose
Data Processing	PCAP Collection	ZDNS
	Hadoop Parsing	Miek Gieben DNS lib
		Kafka
Storage	Apache Avro	Apache Parquet
Schema	Figure 3.2 (p. 30)	Figure 4.2b (p. 71)

index (i.e., `LXC-0` reads `split-0`, `LXC-1` reads `split-1`, `LXC-N` reads `split-N`). This scheme made scaling the system particularly difficult and required significant manual effort in order to rearrange splits for potential changed.

To tackle this limitation, we introduced RabbitMQ [87], which allows for a *producer-consumer* model in a distributed system. Therefore, the only change from our side, had to do with populating the queue and making sure that worker nodes now read from the queue instead of a static file. Using this approach, worker nodes can come and go without affecting the domains that will eventually be resolved. In the case where a large number of workers disappears, we still have a potential bottleneck that might affect speed, but with our current redundancy, we can tolerate more than half of the servers to shut down, without missing the opportunity to query every domain in our seed at least once a day.

Moreover, the queue allows us to arbitrarily accept incoming requests for domains to be added to the queue. Thus, instead of having to coordinate with data providers, we can either provide them with the capability to push data to the queue, or provide us the data and we can run a small application that appends it to our FIFO queue.

4.4.2 Resource Orchestration

Another component that needed to be updated was the reliance on LXC containers. At the time of writing, LXC (and the updated LXD) is a technology that lacks significantly behind

Docker. Unfortunately, our need for VLANs did not allow us to experiment with Docker until 2017, when the `macvlan` network type became available. However, Docker has been extensively updated with a plethora of new features added. Some of those features, like *Docker Compose*, make management of containers extremely easy, via templates and images that are immediately loaded and managed by the Docker service, instead of third party software.

Once our services had been moved to Docker, we were able to take advantage of Docker Compose to manage our services. Docker Compose makes scaling Docker containers trivial, and allows us to scale our lookup capacity rapidly as needed based on consumption rates in RabbitMQ. Docker is also supported by more modern solutions, like Kubernetes, which is a complete solution for automating application deployment, scaling, and management. The introduction of Docker in Thales 2.0 will allow for further exploration of newer technologies, like Kubernetes, to continue scaling and providing the Active DNS data to the community.

4.4.3 Data Processing

The next part of the system, after generating the seed and managing the containerized workers that will be submitting DNS requests, is the core component that will perform the DNS resolution. As mentioned earlier, Thales used to be taking advantage of a combination of our own code to submit the asynchronous resolution requests and Unbound for the recursion. After switching to ZDNS we did not only remove Unbound from the containers themselves, but could also now collect the responses at the container, process them, and then ship them to Kafka, the next component in the ETL pipeline.

The PCAP collection was problematic for the variety of reasons mentioned earlier, in Section 4.3.3, and moving away from it was a focal point of the redesigned architecture. ZDNS gives us the ability to handle responses to queries and manipulate them as they come in. We leverage this to serialize responses directly to Avro in the same application

(the worker) that is performing the resolution requests. Therefore, collecting PCAPs is no longer necessary, and we generate the updated schema.

ZDNS is very lightweight and actively maintained by a private company and the open source community. The ZDNS project has been accommodating of the Active DNS project and not only has the ZDNS project accepted our modifications to better the source code, but have also modified ZDNS to conform to our use case on certain occasions. Among these changes, we have made the input and output of ZDNS modular, so that a plugin could be added that would send data directly to Kafka, instead of writing it to a file.

Once we moved away from processing PCAPs we knew we would need a central component to the system that could temporarily store parsed results before they can be moved to HDFS. Kafka was an ideal solution that fitted our use case as it is Hadoop native, widely supported, and it gives us fault-tolerance in the event of network connectivity issues or other hardware failures that may arise. Therefore, we built a Kafka cluster that is used to temporarily aggregate data from workers and store it until HDFS is ready to consume it.

The combination of ZDNS, Kafka, and RabbitMQ, gives us the ability to horizontally scale our lookup capacity, much easier than before. In Thales we needed to manually configure new hosts added to the system and take care of functional changes (e.g., seed splits, PCAP collection distribution, etc.). With Thales 2.0, we can immediately add a worker node, which will automatically subscribe to both RabbitMQ and Kafka to obtain domains it should query and a location to store the output.

4.4.4 Long-Term Storage

The new architecture takes advantage of Avro when pushing data to HDFS, as mentioned earlier. The workers will push Avro data to Kafka, which will then be transferred to HDFS in the same format. Avro, being Hadoop native, can be processed much faster than the PCAPs we used to process with Thales. Hence, we take advantage of Spark to transform the ingested Avro data into a long-term storage schema and format that will be maintained

for the lifetime of the Active DNS project. Moreover, we utilize the Kafka HDFS Connector from Confluent, which uses a Schema Registry to validate data in transit. Therefore, potentially malformed records that find themselves in Kafka will be excluded from the transfer and not end up in HDFS. This allows for a much more consistent intermediate data, which would not cause problems when converting to our permanent storage format, or cause issues later when working with the Active DNS data.

Parquet, a columnar storage format, provides two significant benefits for long-term storage: (1) higher compression rate, since it will remove duplicate values in partitions, and (2) higher access speed, because it can provide only the relevant data for a given query. However, in order to create a Parquet partition, much more processing capacity is required, since the data needs to be loaded in memory for deduplication, repartitioning, and reformatting. This is an one-time cost, though, that will be paid once the Avro data is in the Hadoop cluster. From that point forward, any processing of the Parquet data is going to be much faster and can be done over larger amounts of data, than in the case of Avro. Moreover, the increased compression rate, allows us to store more information for the same space complexity that we used to tolerate with Avro, which results in more complete context around resource records we store.

4.4.5 Schema

As mentioned earlier, Parquet allows us to keep more columns in the data we store long-term. Therefore, we are able to solve issues we had with the older schema, like remove fields that had very low value for the research community and add ones that will provide more context and allow for validation and potentially enable other research perspectives.

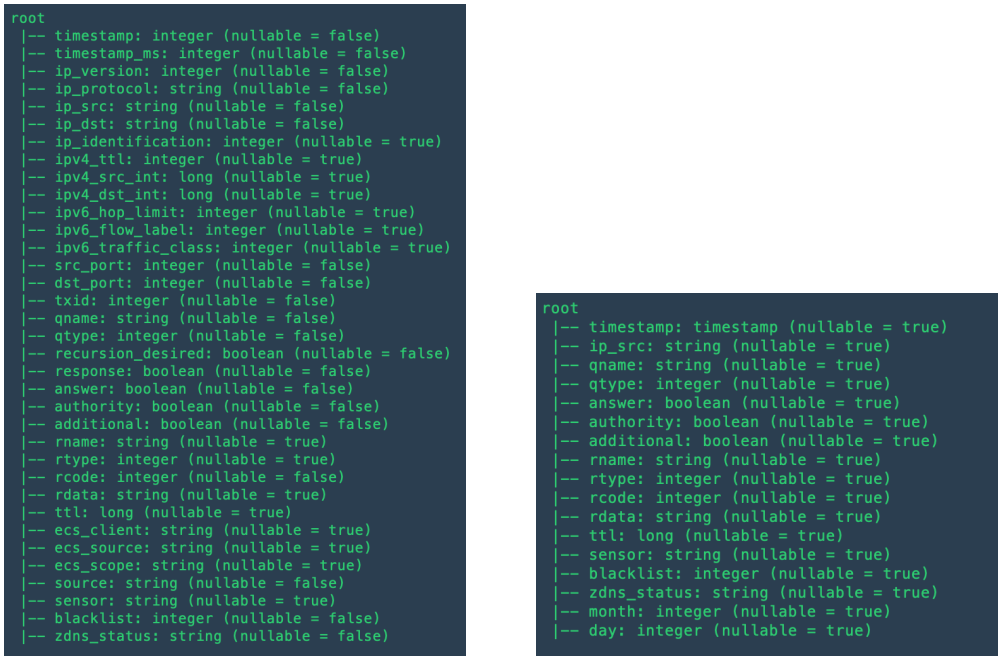
At this point, we should note that there are two effective schemas that Active DNS data is stored into. The first one, is the Avro schema generated by the worker nodes, temporarily stored in Kafka, then temporarily stored in HDFS. This schema is detailed in Figure 4.2a. Data in the Avro schema is maintained in Kafka for up to seven days, and

on HDFS for approximately three days, for redundancy purposes. Therefore, there is a rolling window of seven days available of raw Active DNS data, before any processing has taken place. Moreover, the temporary schema thoroughly details information regarding the DNS resolution process, which, if found necessary, can be migrated to the compact Parquet schema we store permanently.

The compact and permanently stored Parquet schema can be seen in Figure 4.2b. This schema includes most of the information available in the older schema (Figure 3.2, p. 30), along with several more fields. For example, the `answer`, `authority`, and `additional` denote the section of the packet the RR came from, the `qname` and `qtype` are now separate from the `rname` and `rtype` for the user to know what domain was actually queried when the RR was received, and the `rcode` that will provide information about what the authority said regarding the queried domain name. Therefore, more information is now available for the research community, once they obtain the data in the newer schema. For consistency purposes and continuity, we have decided to support both schemas for the foreseeable future, so that Active DNS data consumers will not have to migrate systems that are already running on the older schema. Given that the new schema is a superset of the older one, we are able to do so without any modifications to Thales 2.0.

4.5 Thales 2.0 Value

Changes to Thales described so far have a significant impact on the overall performance of the system as expected. The new system, Thales 2.0, provides us with a significant amount of more information around the Internet Protocol and infrastructure used online. This section presents a series of measurements that demonstrate the value added to the Active DNS dataset after Thales 2.0 started collecting data. We will focus on a recent time period, between December 1st, 2019 and May 31st, 2020, where we have three overlapping datasets: (1) Active DNS collected by Thales (also referred to as *old Active DNS*), (2) Active DNS collected by Thales 2.0 (also referred to as *new Active DNS*), and (3) Passive



(a) (b)

Figure 4.2: Active DNS data schemas. (a) The temporary Avro schema used by the collection system. (b) The long-term Parquet schema the data is stored into.

DNS from a large University. This will shed light into the differences between passive and actively collected DNS data, similar to the analysis presented in Section 3.3.1, p. 34).

4.5.1 DNS Data

Similarly to the former Active DNS data collection system, Thales, Thales 2.0 is resolving millions of domain names on a daily basis, attempting to reach at least two successful resolutions for every QTYPE we are interested into, per day. This translates into approximately four billion resolution requests, for almost 400 million domains per day.

Figure 4.3 shows the daily distribution of the number of unique resource records (RRs) collected by Thales 2.0, Thales, and the passive DNS dataset. The top portion of the two plots uses a logarithmic scale for the y-axis, to make the difference between Active DNS and passive DNS apparent. On any given day, old Active DNS is approximately two orders of magnitude larger than passive DNS, whereas the new Active DNS is at least two orders of magnitude larger, for the six month period we are investigating.

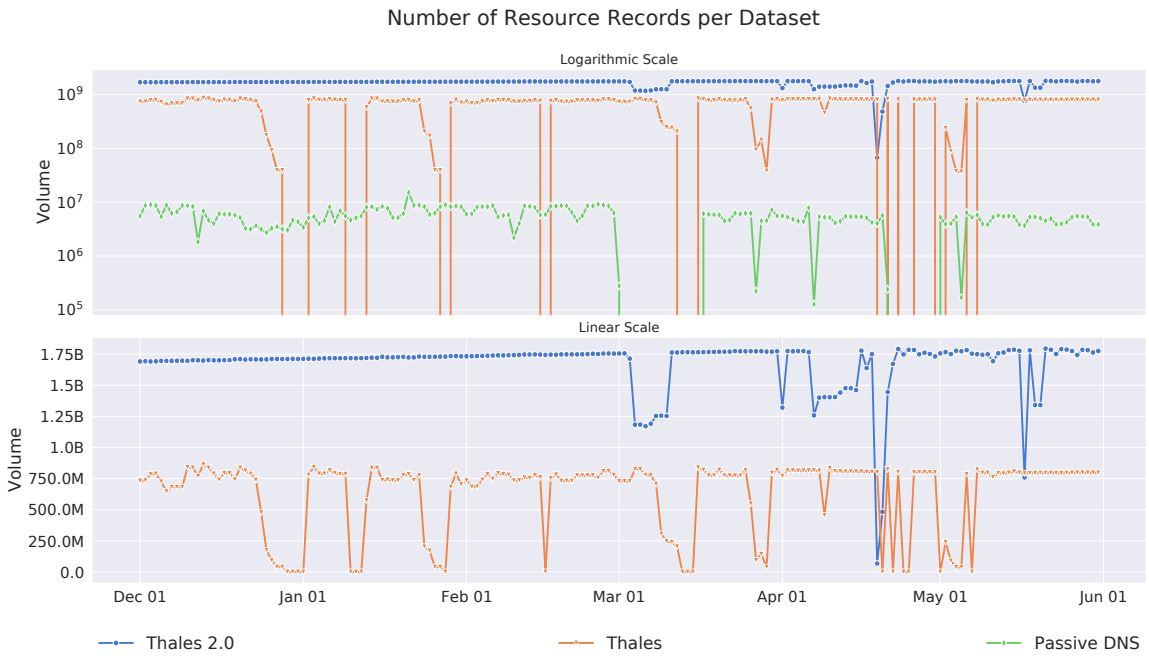


Figure 4.3: Number of Resource Records (RRs) in the former Active DNS dataset from Thales, the new Active DNS dataset from Thales 2.0, and passive DNS dataset from a large University. The top portion of the plot compares the three datasets in logarithimic scale, whereas the bottom presents the difference between the data collected by Thales, and Thales 2.0, in linear scale that better depicts the differences.

Before we go in any deeper analysis, we should explain the significant dips in the plots. Firstly, regarding the new Active DNS, we can see a drop in the RRs collected in early March. Starting March 2nd and for the five following days, a brief network outage caused some of our seed lists to not be updated daily, therefore, the system was only using the available seed data to perform queries, hence, the drop in the dataset. In mid-March, Georgia Institute of Technology moved to a completely online lecture system, for all graduate and undergraduate programs, due to the SARS-CoV-2 pandemic of 2020. We can see that Thales 2.0 remained stable throughout March, but, after Spring Break for Spring 2020, and students restarted with online classes, several network issues that Georgia Tech had faced throughout the online transition have made the system fairly unstable. Given that Thales 2.0 is housed at Georgia Tech's datacenter and shares the same upstream Internet, this was an expected side-effect. However, the way the system has been designed, we

can add more worker nodes to share the load in different networks, which should provide adequate Quality of Service. Nevertheless, we can see that Thales 2.0 is still able to collect data on a daily basis with very high fidelity, even during cataclysmic events, like the COVID pandemic.

With respect to the old Active DNS and Thales, we can notice approximately eight different events where data was not collected for one to five consecutive days. These random events, which started occurring much more often after March 2020, depict one of the reasons why we had to migrate to Thales 2.0. When something would go wrong with Thales, we needed to manually look into it and fix the problem. What we see in this plot are cases of network outages, power outages, and the lag before a human intervened to bring the system back online. We can see similar problems with Thales 2.0, however, there is no day for which Thales 2.0 collected no data at all. On the contrary, even when a network outage caused the seed to be incomplete (early-March), the system did collect data, demonstrating the availability capabilities of Thales 2.0.

Finally, the same network outage in early March caused the passive DNS data collection from the campus to be incomplete. Moreover, the increased bandwidth usage from the online transition for Georgia Tech is also apparent after March, where we see larger fluctuation. Moreover, we can see that the passively collected DNS data keeps declining between February and June 2020. This is an inherent problem that passive DNS data has, since in order to collect DNS data passively, it needs to be generated by a device in the network. Even though students moved to online classes and were using a Virtual Private Network (VPN) for most of their academic work, almost every building at Georgia Tech was in lockdown, meaning that computers, equipment, and Internet-connected systems were no longer generating traffic. Hence, the steady decrease in network traffic, led to a decrease in DNS data collected, with the most significant reduction during the Summer semester, after May 1st.

The bottom portion of Figure 4.3 depicts the difference between the old and new Active

DNS datasets in a linear scale to make it more comprehensive. As we can see in that plot, Thales 2.0 (blue) presents an upward trend in unique RRs collected on a daily basis, whereas Thales seems to be almost stable. Also, we can see that the new Active DNS datasets contains approximately one billion RRs more than the older Active DNS dataset. There are two explanations for this discrepancy. First, the new Active DNS has a much more robust way of updating the seed of domains that will be queried daily; therefore, the new seed includes more domains than the older one, which come from sources subscribed directly to the RabbitMQ queue. Second, the old Active DNS dataset contains a plethora of NXDOMAINs that come from domains that have been removed from the current zonefiles. In other words, Thales has been using a concept similar to an append-only list for the seed, which would inevitably include expired and deleted domains, that Thales 2.0 ignores in an attempt to optimize performance. Thus, querying a domain that does not exist will generate a single RR (`QNAME IN A NXDOMAIN`), whereas querying a domain that does exist will generate from five RRs (one for each QTYPE), to as many as the different RDATA the domain owner has set.

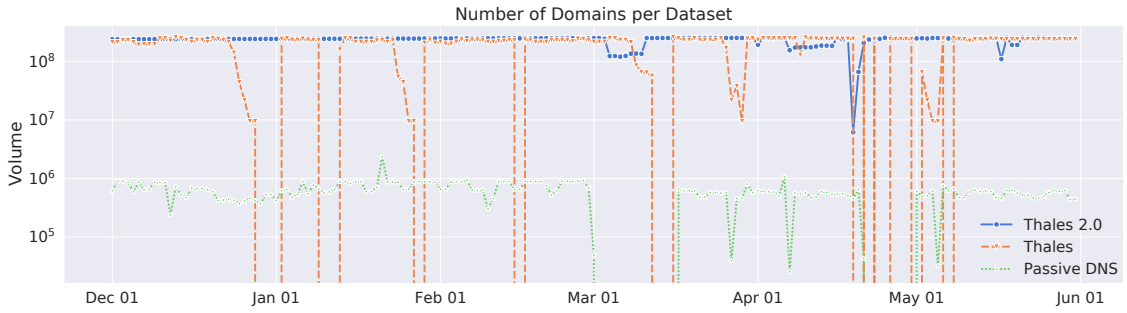
Similarly, Figure 4.4 demonstrates statistics in the DNS data from all three sources (blue for Active DNS collected by Thales 2.0, orange for Active DNS collected by Thales, and green for passive DNS). The number of domain names resolved and number of registered domains (e2LDs) in Figure 4.4a and 4.4b are almost the same for both Active DNS datasets, but significantly more than the passive DNS dataset. We can see at least three orders of magnitude more domains and e2LDs in the Active datasets compared to the passive one (note that the y-axis is in logarithmic scale for all plots), in line with what we saw four years ago in Figures 3.5a and 3.5e, p. 36.

On the other hand, we can see that Thales 2.0 collects a larger amount of IP addresses and significantly larger RDATA overall, in Figures 4.4c and 4.4d respectively. As mentioned earlier, the NXDOMAINs and potential mismatch of the seed between the two systems, with Thales 2.0 resolving more domains than Thales, can easily explain why the new

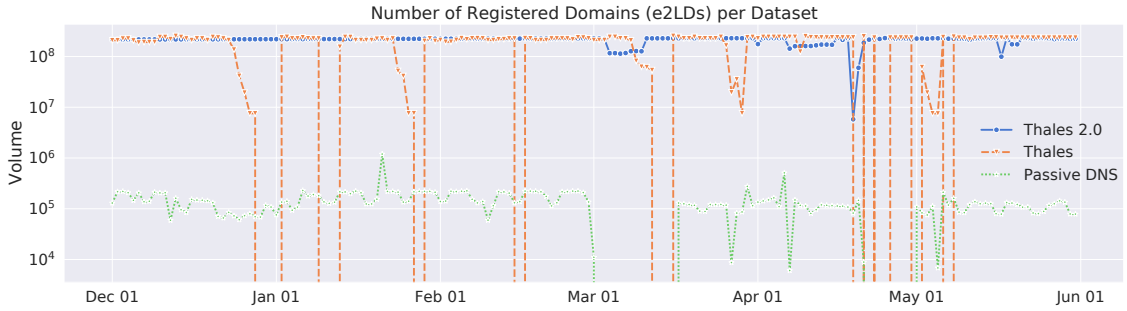
Active DNS dataset can create a more complete view of the Internet infrastructure used to host domain names. At the same time, the increase in RDATA values allows for more fine-grain research into different data that the security community would need, like `TXT`, or `AAAA` records.

Table 4.2 provides a breakdown of the underlying data in each of the three datasets described so far (new Active DNS, old Active DNS, and passive DNS). Note that the values in the table are in thousands. We can see that the number of domains is very close for the Thales 2.0 and Thales, but almost 500 times more than in passive DNS. The same is true for Resource Records (RRs), which can be explained by the large discrepancy in the RDATA collected in each dataset. As we can see, the new Active DNS dataset includes at least twice as many data points in the RDATA section than the older Active DNS dataset and almost 55 times more than passive DNS. Similarly, Table 4.3 depicts the volume of IPv4 and IPv6 addresses in the datasets, the volume of registered domains (e2LDs), and the graph density of the undirected bipartite graph created from the RRs in each dataset. We can see that, unlike the overall RDATA, when it comes to IP addresses, the new Active DNS has approximately 4M more addresses than the older one, but 16M more than passive DNS. The same stands for registered domains (e2LDs) where we see that the new Active DNS has approximately seven to 10 million more domains than the older dataset, but almost 200M more than passive DNS.

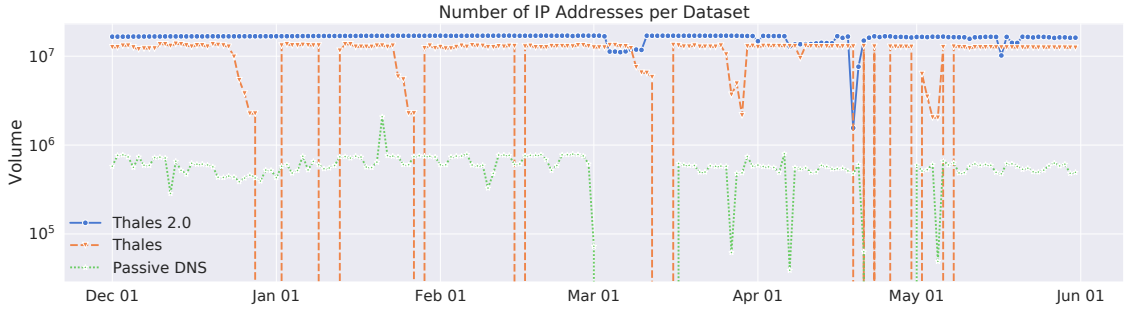
Overall, we can observe that Active DNS provides a way larger *breadth* of data than passive DNS, with Thales 2.0 yielding even more data points, but much less *depth* in each zone observed. This is easily noticeable if we take December 1st for example; the new Active DNS dataset has approximately 240M domains and 215M registered domains (e2LDs), leaving just 35M repeated domains (hence, child labels in different zones). On the contrary, passive DNS has 616K domain names, only 130K of which are unique e2LDs. Therefore, we can see that almost 80% of the FQDNs in passive DNS are domains under a zone already observed, compared to just 10% in the new Active DNS. This is also apparent in the



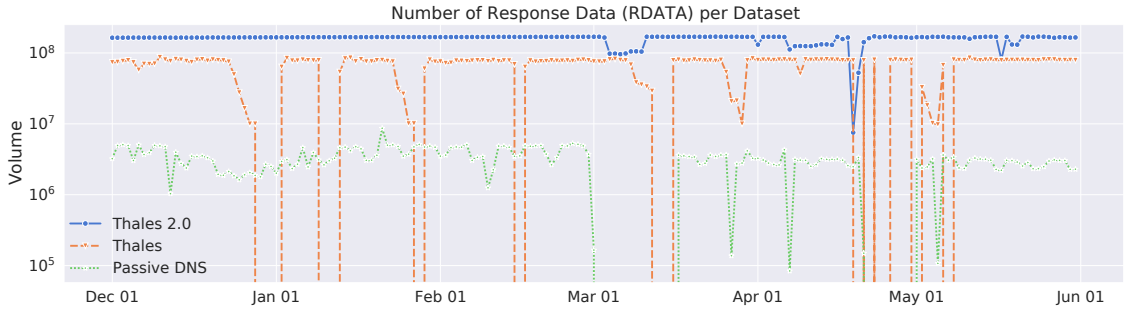
(a) The number of unique domain names in each of the datasets discussed.



(b) The number of unique registered domains (e2LDs) in each of the datasets discussed.



(c) The number of unique IP addresses in each of the datasets discussed.



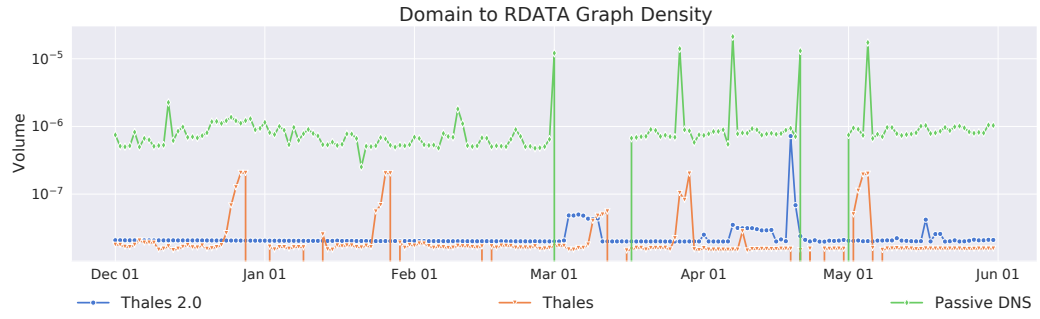
(d) The number of unique Response Data (RDATA) in each of the datasets discussed.

Figure 4.4: Comparison of data points in the previous Active DNS dataset collected by Thales, the newer Active DNS dataset collected by Thales 2.0, and a passive DNS dataset from a large University.

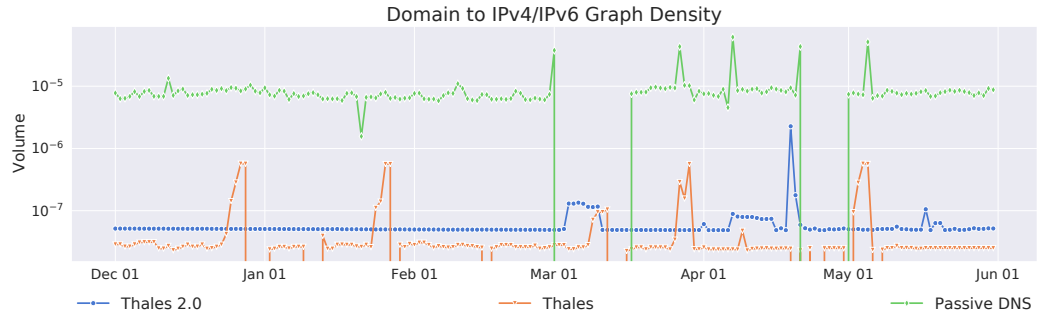
graph density we can see in Table 4.3 . The density in passive DNS is much larger than the density we see in Active DNS. This is expected since the graph density is provided by the formula:

$$D = \frac{|E|}{\binom{|V|}{2}} = \frac{2|E|}{|V|(|V|-1)}$$

where E is the number of edges, or RRs in our case, and V the number of nodes, or RDATA and domains. Therefore, as the number of domains queried, which are not related to other domains previously queried, grows, the density of the graph will drop. This is what we are experiencing with Active DNS, thus, we can conclude that we have a much larger and much more sparse graph in Active DNS than passive DNS. This is also reflected in Figures 4.5a and 4.5b, where we can see the daily graph density of domain to RDATA (RRs) and domain to IPv4/IPv6 (RRs for `A` and `AAAA` records), respectively.



(a) RDATA graph density.



(b) IPv4/IPv6 graph density.

Figure 4.5: Density of the bipartite graph of domain name to RDATA (a) and domain name to IPv4 and IPv6 (b).

Table 4.2: Number of domains, RDATA, and Resource Records (RRs), collected over the first two weeks of December 2020. Values are in thousands ($\times 10^3$).

Date	**Domains**			**RDATA**			**RRs**		
	Thales 2.0	**Thales**	**Passive DNS**	**Thales 2.0**	**Thales**	**Passive DNS**	**Thales 2.0**	**Thales**	**Passive DNS**
Dec 01	239,277	212,858	616	164,000	73,590	3,229	1,691,274	731,712	5,536
Dec 02	239,365	212,975	898	164,231	74,492	4,944	1,692,965	733,932	8,722
Dec 03	239,381	229,907	899	164,294	76,790	5,088	1,691,233	784,048	8,961
Dec 04	239,500	230,586	858	164,494	78,701	4,944	1,692,724	788,245	8,704
Dec 05	239,740	211,758	590	164,750	73,619	3,037	1,695,348	725,264	5,416
Dec 06	239,968	192,641	863	164,784	57,725	5,069	1,695,863	646,096	8,757
Dec 07	240,070	196,583	639	164,715	69,924	3,692	1,696,155	680,977	6,241
Dec 08	240,160	196,526	653	164,808	69,632	3,920	1,696,732	680,782	6,622
Dec 09	240,185	196,580	863	164,981	70,510	4,969	1,697,336	682,088	8,656
Dec 10	240,056	246,473	855	164,611	87,156	4,899	1,696,501	840,471	8,627
Dec 11	240,299	246,973	843	165,273	80,535	4,773	1,701,362	837,063	8,343
Dec 12	240,591	224,592	238	164,603	76,762	1,036	1,700,422	769,859	1,836
Dec 13	240,388	258,687	727	164,424	81,901	3,975	1,698,599	862,807	6,821
Dec 14	240,833	247,109	528	164,947	79,972	2,764	1,701,880	835,679	4,606

Table 4.3: Number of IPv4/IPv6 addresses, number of registered domains (e2LDs), and RR graph density for the first two weeks of December 2020. Values are in thousands ($\times 10^3$).

Date	**IPv4/IPv6**			**e2LDs**			**RR Density**		
	Thales 2.0	**Thales**	**Passive DNS**	**Thales 2.0**	**Thales**	**Passive DNS**	**Thales 2.0**	**Thales**	**Passive DNS**
Dec 01	16,575	12,580	578	215,855	208,065	130	2.08×10^{-8}	1.78×10^{-8}	7.49×10^{-7}
Dec 02	16,588	12,598	765	215,943	208,179	217	2.08×10^{-8}	1.78×10^{-8}	5.11×10^{-7}
Dec 03	16,586	13,092	776	215,979	224,920	220	2.08×10^{-8}	1.67×10^{-8}	5.00×10^{-7}
Dec 04	16,591	13,112	735	216,087	225,564	207	2.07×10^{-8}	1.65×10^{-8}	5.17×10^{-7}
Dec 05	16,626	12,537	562	216,293	206,957	146	2.07×10^{-8}	1.78×10^{-8}	8.24×10^{-7}
Dec 06	16,634	11,944	738	216,518	188,216	206	2.07×10^{-8}	2.06×10^{-8}	4.98×10^{-7}
Dec 07	16,646	12,182	593	216,629	191,928	138	2.07×10^{-8}	1.92×10^{-8}	6.66×10^{-7}
Dec 08	16,653	12,179	591	216,718	191,888	140	2.07×10^{-8}	1.92×10^{-8}	6.33×10^{-7}
Dec 09	16,660	12,196	723	216,743	191,936	212	2.07×10^{-8}	1.91×10^{-8}	5.09×10^{-7}
Dec 10	16,661	13,585	731	216,644	241,291	205	2.07×10^{-8}	1.51×10^{-8}	5.21×10^{-7}
Dec 11	16,687	13,603	715	216,868	241,780	207	2.07×10^{-8}	1.56×10^{-8}	5.29×10^{-7}
Dec 12	16,703	13,030	285	217,134	219,602	59	2.07×10^{-8}	1.70×10^{-8}	2.26×10^{-6}
Dec 13	16,691	13,758	658	216,950	253,403	161	2.07×10^{-8}	1.49×10^{-8}	6.17×10^{-7}
Dec 14	16,719	13,616	520	217,359	241,903	96	2.07×10^{-8}	1.56×10^{-8}	8.50×10^{-7}

Taking a closer look at the RDATA collected by Active DNS, allows us to paint a clear picture of the IP infrastructure we can observe. As mentioned earlier, we are able to retrieve more than 16 million IP addresses on a daily basis from around the world. If we focus on IPv4, which is the most widely used DNS record, we can identify the Autonomous System Numbers (ASNs) and Owners (ASNames) that announce each IP address, the Border Gateway Protocol (BGP) prefix announced for route selection, and the country each IP is in.

Table 4.4, presents a detailed comparison of the aforementioned data points for Thales and Thales 2.0 throughout the six months we have been looking into so far (December 1st, 2019 through May 31st, 2020). As we saw in Figure 4.4c, Thales 2.0 is able to collect more IP addresses than Thales, which translates in almost 100,000 more BGB prefixes announced, from approximately 8,500 more ASes, which belong to 7,800 more organizations (AS names). These IP addresses yield visibility to two more countries than what we were able to identify in Thales. A very similar distribution is apparent if we look at just one day, December 1st, 2019, which is available in Table 4.6, for reference. Moreover, Table 4.5 presents the percent coverage of the two different systems. We can see that Thales 2.0 provides approximately 0.2% more coverage in IPv4 addresses, 11% more coverage in BGP prefixes, 11% more coverage in ASNs, 11% more coverage in ASNames, and 1% more coverage when it comes to countries identified. The tables also include the corresponding values for the passive DNS dataset over the same time period, to provide an overall comparison.

At this point, we should note that the numbers in the three tables (Table 4.4, 4.5, and 4.6) only include non-*bogon* IP addresses. Bogons [107] are IP addresses that are *reserved* (e.g., RFC 1918 [74], RFC 5735 [75], RFC 6598 [76]), but have not been *allocated* or *assigned* by the Internet Assigned Numbers Authority (IANA), or any Regional Internet Registry (RIR).

The three maps in Figure 4.6 depict the geographic location of the IP addresses in the

Table 4.4: Detailed IPWHOIS and BGP information for the RDATA in A records, as seen in Active and passive DNS for six months of data, from 2019-12-01 through 2020-05-31. Values in this table exclude Bogon [107] IP addresses.

	IP Addresses	Prefixes	ASNs	ASNames	Countries
Total in IPv4	3.96B	863K	67,345	65,363	237
Thales	20,432K	330K	46,838	45,852	235
Thales 2.0	28,806K	427K	54,346	53,079	237
Passive DNS	6,923K	252K	44,271	43,454	236

Table 4.5: Percent coverage of IPWHOIS and BGP information for the RDATA in A records, as seen in Active and passive DNS for six months of data, from 2019-12-01 through 2020-05-31. This table is the result of Table 4.4. Values in this table exclude Bogon [107] IP addresses.

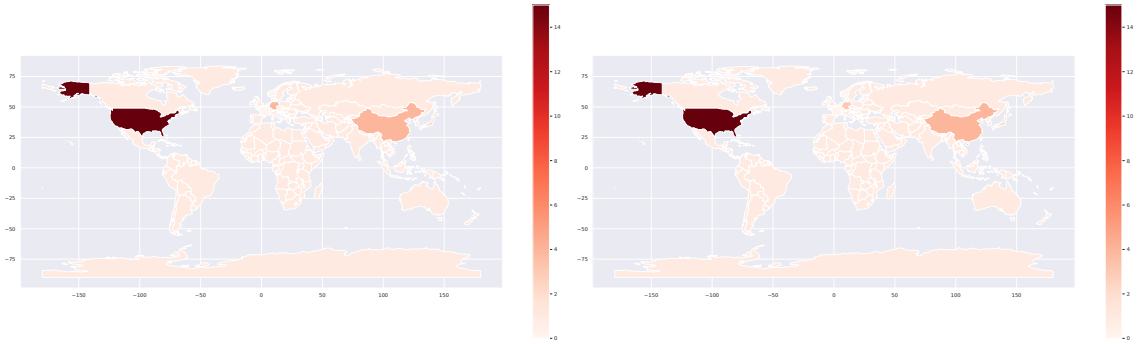
	IPv4 Addresses	Prefixes	ASNs	ASNames	Countries
Thales	0.52%	38.23%	69.55%	70.15%	99.16%
Thales 2.0	0.73%	49.51%	80.55%	81.21%	100%
Passive DNS	0.17%	29.16%	65.74%	66.48%	99.58%

three datasets, new Active DNS, old Active DNS, and passive DNS, in that order. As we can see, all three datasets have a very similar distribution in terms of countries that can be identified in A records. However, both Active DNS datasets have many more IP addresses in each country, with China and the Netherlands demonstrating the most significant difference. Among the two Active DNS datasets, we can see that Thales 2.0 presents a few more million IP addresses overall, but the difference is not much larger than Thales.

Finally, Figure 4.7, provides a breakdown of the volume of data for each individual QTYPE we query for and potential RTYPEs we receive during the resolution process (i.e., NS, and CNAME). We query for five different QTYPEs, A, AAAA, MX, SOA, and TXT, and we can see that Thales 2.0 is able to collect much more data points than Thales for every QTYPE (AAAA — Figure 4.7b, MX — Figure 4.7d, SOA — Figure 4.7f, and TXT — Figure 4.7g), but almost the same number for A records (Figure 4.7a), inline with what we saw earlier on the daily data collection tables (Table 4.2, and 4.3). The two QTYPEs that are collected "for free", NS and CNAME, given they are part of the resolution process, are

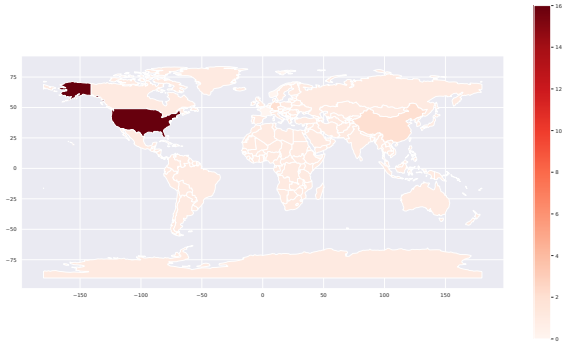
Table 4.6: Detailed IPWHOIS and BGP information, similar to Table 4.4, but for December 1st, 2019 only. Values in this table exclude Bogon [107] IP addresses.

	IPv4 Addresses	Prefixes	ASNs	ASNames	Countries
Total in IPv4	3.96B	863K	67,345	65,363	237
Thales	12,492K	252K	42,813	42,015	235
Thales 2.0	16,460K	304K	48,200	47,244	235
Passive DNS	575.6K	64.7K	14,935	14,656	204



(a) Thales 2.0 Active DNS.

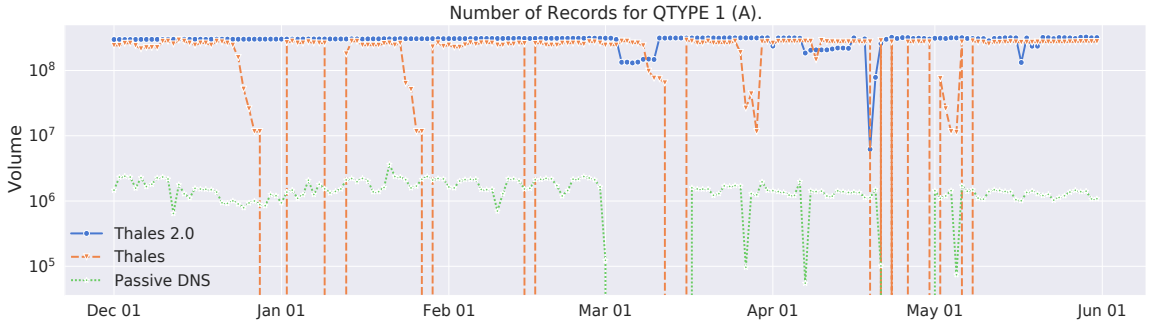
(b) Thales Active DNS.



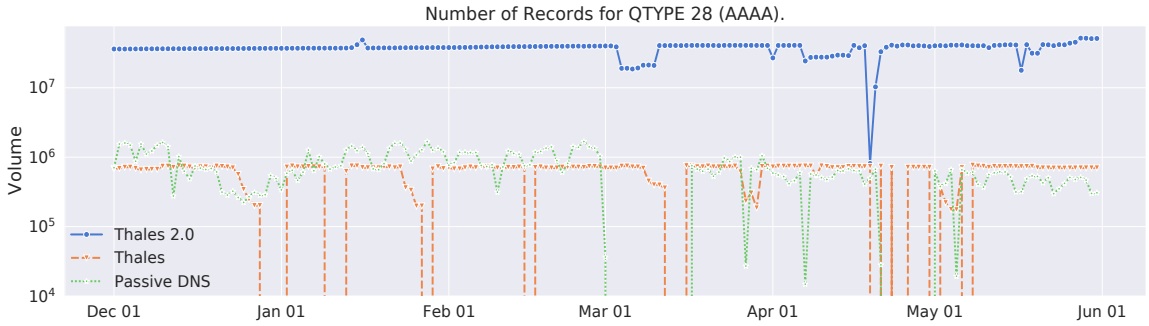
(c) Passive DNS.

Figure 4.6: Geographic distribution of the IP addresses found in the three different datasets.

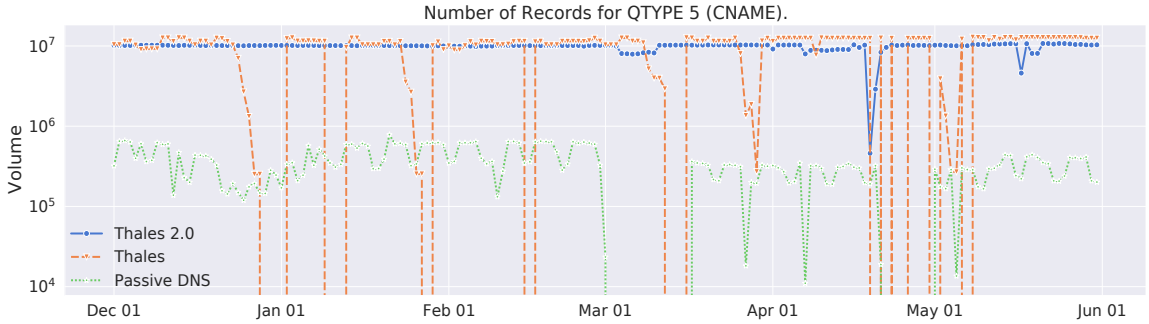
almost the same for both datasets. This is expected since the number of domains queried has not significantly changed and many domains from different TLDs are represented in both systems. Hence, it is reasonable to have adequate visibility in the DNS authorities that serve domain names around the world. Moreover, `CNAME` records are very common in CDNs and, as we have seen, our visibility into the most popular TLDs (e.g., `com`), provides extensive information about popular domains hosted in CDNs. Therefore, it is reasonable to expect to see a very similar distribution between the two systems in terms of `CNAME`



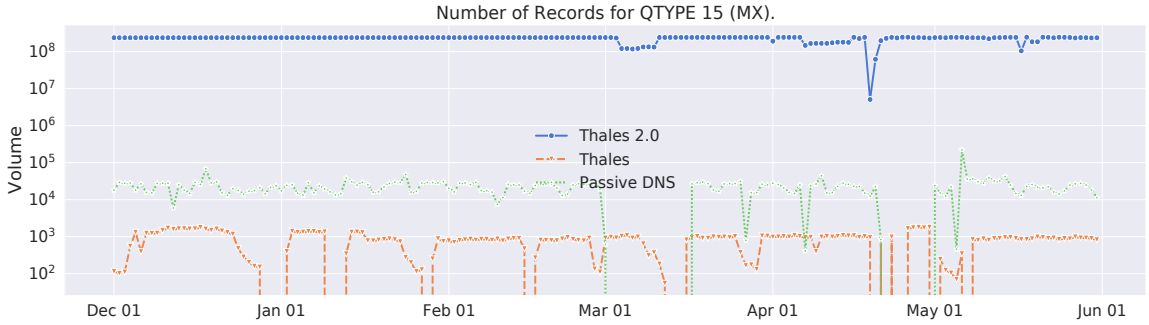
(a) A.



(b) AAAA.

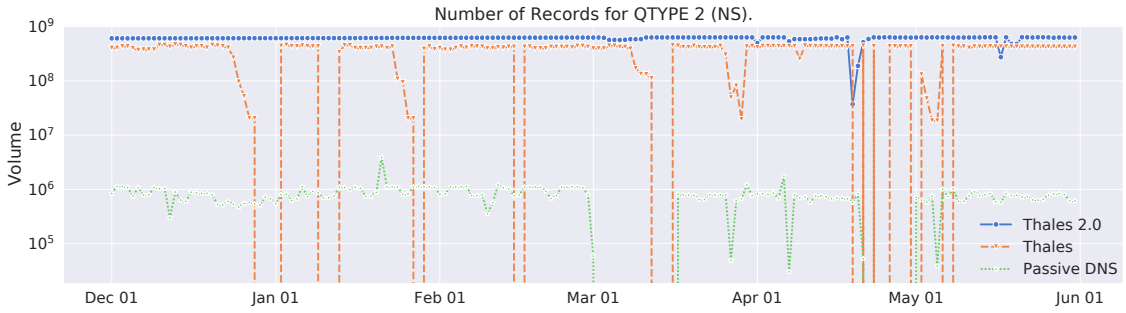


(c) CNAME.

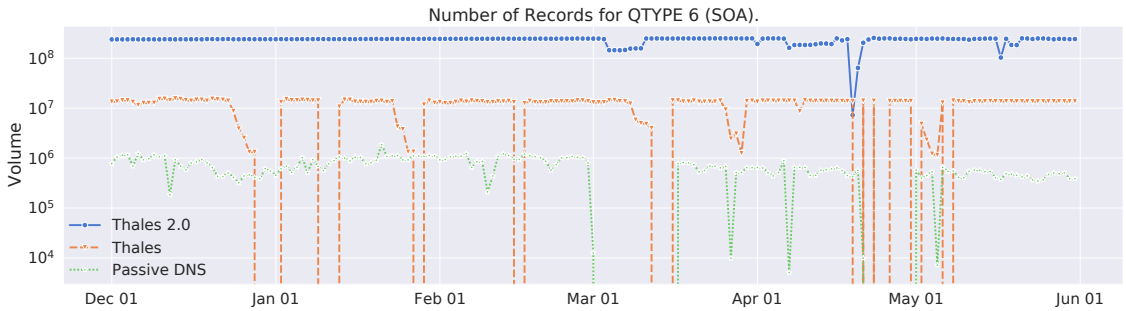


(d) MX.

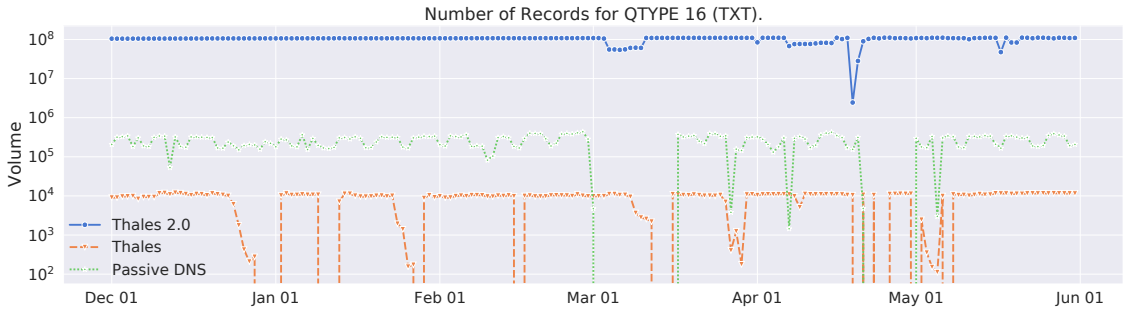
Figure 4.7: Number of records in each dataset for the QTYPEs resolved.



(e) NS.



(f) SOA.



(g) TXT.

Figure 4.7: Number of records in each dataset for the QTYPEs resolved (continued).

records collected.

On the contrary, passive DNS data has a much lower volume in every QTYPE, ranging from three (in most QTYPEs) to five orders of magnitude for MX records. This is also expected, since passively collected DNS requires DNS queries to be submitted by systems in a network where collection is taking place. User stimulated DNS resolutions, or system applications communicating on the Internet, will be the only packets collected, significantly limiting the data available. For example, in the case of MX records, the resolution will only be visible if someone is attempting to perform an MX query, which is the logical sequence

of an attempt to send an e-mail. It would be very rare for an application to resolve an `MX` record if it is not participating in the sending, receiving, or relaying of e-mail messages. This is where Active DNS is particularly powerful. It does not depend on any user or system activity to collect such records, therefore providing a more complete view of the records available for the domains Active DNS has visibility into. Thus, the more domains Thales 2.0 can query, the more data points we can collect on a daily basis and enhance the resulting dataset. Changes to the original system allow us now to incorporate many different data sources and collection nodes (vantage points), which can allow for the resulting dataset to scale even further.

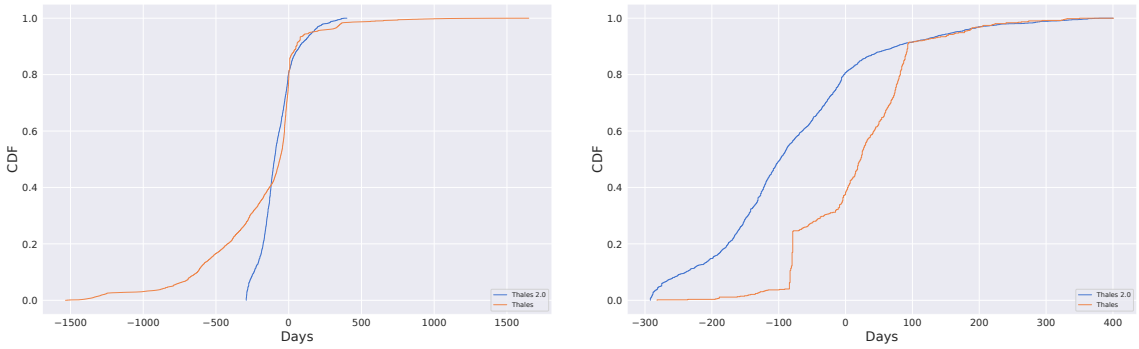
4.5.2 Active DNS Data in Security Research

So far we have discussed how Thales 2.0 has contributed in the DNS data collected to compose the Active DNS dataset. This section describes how the increased data volume and system reliability and performance can impact security research through enhancing blacklists and malware DNS datasets, similar to Section 3.4.1, p. 40. As we will see, Thales 2.0 has the potential to assist the security community even further than Thales did.

Public Blacklist

Blacklist information, part of Open Source Intelligence (OSINT), is often used by the security community for various reasons, both operationally (e.g., as a blocklist of domains or IPs a device should not contact), and in research and development (e.g., as a means to train and evaluate a detection system). As we showed earlier, in Section 3.4.1, p. 40, the Active DNS data collected by Thales has had visibility into malicious domain names several days, even months, before they are identified and released in a malicious domains list.

In order to further demonstrate the advantages of upgrading Thales to Thales 2.0, we perform the same experiment as we did almost five years ago. We collect malicious domains that appear in public blacklists and then indentify them in both Active DNS datasets



(a) (b)

Figure 4.8: The number of days before a blacklisted domain name is found in the old and new Active DNS before it is identified in a blacklist. The plot on the left (a) includes every domain name in the last five years Thales has been running for, whereas the plot on the right (b) includes only domains from the overlapping time period of the last year.

collected by both systems. Figure 4.8 depicts the time before (or after) a domain name is found in the Active DNS datasets, versus the blacklist. Similar to Figure 3.7, p. 42, the x-axis 0 is the day that a domain was found in the blacklist data. Therefore, anything on the left hand side of 0 is found in the Active DNS data before it was found in the blacklist data, whereas anything on the right of 0 is first found in blacklist data. Moreover, Figure 4.8a includes every domain found over the almost five year period Thales has been running for, whereas Figure 4.8b only includes the overlapping time period of almost a year.

In Figure 4.8a, we can see that both systems include more than 60% of the blacklisted domain before they are found in a blacklist. Even though comparing the two systems in this fashion is probably not ideal, since Thales has a four year advantage over Thales 2.0, we can still see that Thales 2.0 outperforms Thales in the overlapping time period for approximately one month (in the x-axis). This is more apparent in Figure 4.8b, where we focus only on the last year of data for both datasets. Here we can see that Thales has been lagging significantly behind Thales 2.0. In fact, we can see approximately 40% of the domain names in Thales 2.0 were seen more than three months before they are ever blacklisted (100 days) and 80% of the domains are found the day they are blacklisted, or before. In contrast, Thales has only seen approximately 25% of the domains about 90 days

earlier, and only 40% the same day or before the blacklisting event.

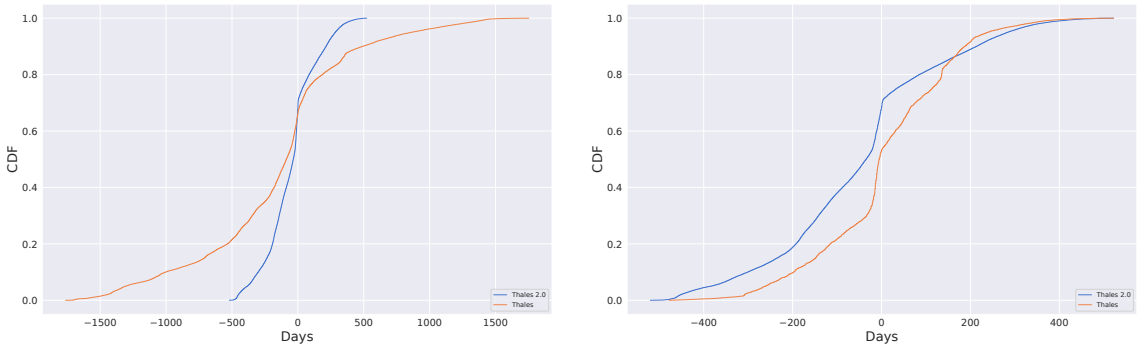
Figures 4.8a and 4.8b show that the Active DNS dataset from Thales 2.0 is significantly richer in domain names and achieves visibility into the blacklisted domains earlier than Thales. Even though, overall, Thales has identified a total of 245K blacklisted domains over five years, it has only found 626 new domains in the overlapping time period. In the same (overlapping) time period, Thales 2.0 has found 2,150 domains. This significant discrepancy, along with the quantitative performance benefits we saw in Section 4.5.1, demonstrates the ability of Thales 2.0 to outperform Thales in the future.

Malware Traces

For several years the security community has been collecting malware samples and dynamically executing them to extract behavioral information and data. Among the artifacts that are obtained, we have been collecting domain names and IP addresses for almost a decade. These domains are often used in the same way as blacklist data that we discussed earlier. We perform a similar analysis to the one we did in Section 4.5.2, but for malware DNS traces. We take domain names that were found in the network traces of dynamically executed malware binaries and we compare the day that the sample was found and executed, versus the day that the domains it used were found in the Active DNS data.

Figure 4.9, similar to Figure 4.8, demonstrates the number of days a domain name is found in the Active DNS datasets from both Thales and Thales 2.0 before (or after) it is found in the malware traces data. Like before, Figure 4.9a includes the domain names from the entire time period Thales has been running for, whereas Figure 4.9b only includes the last year of overlapping data.

In the first case, Figure 4.9a, we can see that in the long-run Thales outperforms Thales 2.0. Approximately 40% of the domains show up in Active DNS from Thales approximately four months before they appear in dynamic execution; for four months, Thales 2.0 can only see approximately 18% of the domains. However, this comparison is, again,



(a) (b)

Figure 4.9: The number of days before a malware domain name is found in the old and new Active DNS before it is identified by dynamic execution of malware binaries. The plot on the left (a) includes every domain name in the last five years Thales has been running for, whereas the plot on the right (b) includes only domains from the overlapping time period of the last year.

not fair and heavily biased towards Thales. Thales had been running for much longer than Thales 2.0, which can explain the increased performance.

However, if we focus only on the overlapping time period, as if both systems were started at the same time, we can see that Thales 2.0 outperforms Thales by at least 20% of the domain names each system has resolved. In fact, Thales 2.0 is able to identify more than 70% of the malware domains before they appear in dynamic execution, when Thales can only do so for approximately 55% of the domains. Looking further back in time, we can see that Thales 2.0 has resolved about 40% of the domains about three months before they appear in dynamic execution traces, whereas Thales has only resolved approximately 25% for the same time frame.

These results demonstrate the ability of Thales 2.0 to outperform Thales long-term. Since Thales 2.0 has been performing better than Thales over the last year, we expect this performance increase to sustain and therefore produce higher quality data for the research community. Overall, Thales has ever seen 730K malware related domains, but only 10,720 in the last year. On the other hand, Thales 2.0 has resolved almost 54,000 domains in the last year alone.

The comparison of Thales and Thales 2.0 overall shows that Thales has performed

better than Thales 2.0 in the last five years. However, when narrowing down the scope of the comparison to only the overlapping time period of the last year, we can see that Thales 2.0 performs significantly better. Hence, we expect that Thales 2.0 will provide higher quality data to researchers and the academic community than Thales did.

4.6 Lessons Learned

The Active DNS dataset has been a significant source of intelligence for the security research community, with more than 70 entities taking advantage of it for their needs. So far, we have described how the data used to be collected, what issues we faced throughout the years and how we tackled them. In this section, we describe the lessons learned after running a distributed system that actively collects DNS data for almost five years.

Human Involvement. For a very long period of time, we have been basing much of the system's fault tolerance and recovery to system administrators and users. Given the technologies available at the time of initial design, it was particularly hard to achieve a level of automation that would have alleviated this burden. As we saw earlier, in Figures 4.3 and 4.4, pp. 72, 76, the lack of automation on disaster recovery has led to brief periods of inactivity. Unfortunately, in most cases, system inactivity explicitly means data loss. In our case, the periods when Thales was offline were not alarming us, since Thales 2.0 was operational at the same time and would collect the data that Thales did not. However, when resources are limited and operators do not have the luxury of running redundant systems in different setups, identifying issues very early can be paramount for data integrity.

Coming to the realization that maintaining Thales was almost a full-time job, made it clear that we had to make radical changes to eliminate multiple points of failure and create infrastructure that can recover from errors automatically. Moving to a more sophisticated code-base and management system, like Docker containers, *docker-compose*, RabbitMQ, and containerized services allowed us to remove humans from the critical path when unforeseen events could cause a temporary interruption to the system.

Bug Elimination from Open Source. The assembly of scripts and (effectively) "glue code" that kept Thales together was another significant vulnerability. Having written many scripts to take care of service deployment, data collection, data aggregation, data pipelines, data processing, and other components of Thales, we quickly came to the conclusion that using open-source software would have been much easier. Given that open-source software is (usually) community based and driven, there are many more sets of eyes that will look at a piece of code, many more entities that will have deployed it, much more time dedicated to it. These are just some of the advantages that come with open-source software, which can rarely be found in code developed in-house by a small group, like a research lab.

As we can see when comparing Thales (Figure 3.1, p. 28) and Thales 2.0 (Figure 4.1, p. 53), there are several components that have radically changed, or swapped, in favor of open-source software. For example, the scripts that managed the seed have been replaced with an intelligent FIFO RabbitMQ queue, the code performing DNS queries and Unbound have been replaced with `zdns`, data collection has been replaced with Kafka, and Spark is now used to write data into the permanent storage format, instead of PCAP parsing with Map-Reduce Hadoop jobs. Replacing LXC containers with Docker-based containers has been another significant improvement, which allows not only for faster development, but also for consistent and transparent deployment, even in remote environments, almost as a "turn-key" solution. Moreover, we expect that Docker is also future-proof, since active development around Docker from some of the largest Information Technology companies (e.g., Google, Amazon, Microsoft, etc.) is flourishing.

Fault Tolerance. As mentioned earlier, making sure that a system can recover from errors as fast as possible is very important, especially when unrecoverable data loss is involved. The former system, Thales, had several points of failure that had to be monitored and validated manually. This is a cumbersome process, which is particularly hard to automate or develop monitoring tools for.

Distributed systems, as the name suggests, have several components that need to be

monitored in order to verify the well-being of the system as a whole. Moreover, there are certain controls in place to avoid points of failure and maintain continuity and operation with high availability. DNS collection, except for the traditional distributed systems issues [108], has other caveats that need to be addressed.

First, DNS works primarily over UDP, hence, we are looking at a stateless communication protocol, which has no notion of error detection or correction. Therefore, developing a system that will submit billions of resolution requests daily, the results to which are not going to be consumed immediately, requires the developer to build certain controls to account for successful resolutions and take action on unsuccessful ones, based on the type of error. For instance, an `RCODE=1`, or `FORMERR`, which denotes that the query submitted was malformed, might mean that the server does not support DNS Extensions (RFC 6891, `EDNS(0)`) [109], or an `RCODE=2`, `SERVFAIL`, could mean that the authority is currently busy and cannot answer to a request at this time, but might be able to respond later. These issues need to be addressed at the Application Layer, or the software that submits the DNS resolution requests to maximize data collection. In our case, we try to account for these errors by querying for every domain name at least twice and at least eight hours between subsequent queries.

Second, submitting DNS resolution requests is equivalent to sending UDP packets to several servers (the DNS hierarchy) until a final response for the initial query name (domain) has been retrieved. As we discussed in Section 2.1.2, p. 9 and Figures 2.2, and 2.3, pp. 10, 11, a series of DNS requests will be sent to the DNS hierarchy, until the recursive receives the IP address of the authority responsible for the particular zone in question. Then the recursive will query that authority for the RDATA of the domain it has to resolve. The recursive will cache intermediate (and the final) responses, to avoid wasting time resolving the IP address of popular TLDs and authorities in the future. One can immediately realize that if the same authority is responsible for more than one domain name, then it will be queried more than once by Thales. In fact, given that we query for five different QTYPEs,

if an authority is responsible for 10 domains, then it will receive at least 50 requests, assuming that there were no response packets lost. Similarly, if the authority is responsible for 10 million domain names, then it will receive 50 million requests. This is usually the case with very popular authorities, like authorities used by GoDaddy, namecheap, Google Cloud Platform, AWS, etc. In fact, during the summer of 2017, we found ourselves being network throttled by GoDaddy, because we were exceeding their Distributed Denial of Service (DDoS) Attack detection threshold. This issue was quickly resolved through excellent communication with GoDaddy's incident response team, but we might have had the same happen with other smaller authorities.

Identifying cases where an authority is blocking DNS resolution requests is particularly hard. Discriminating between intentional traffic blocking, packet loss, or authority downtime, over UDP, is a non-trivial problem. Since Thales, at first, and Thales 2.0 later, could achieve a very high query rate, we had to account for the potential of abusing the receiving end of the requests. Our primary goal is to not overwhelm authorities on the Internet, and not to simply hide our requests. Hence, we did not choose to use different IP addresses and mask our behavior, since that would still be abusive. Instead, we take advantage of the authority server FQDN found in the zonefiles to extract the registered domain (e2LD) and measure the amount of traffic we would send to each authority. This is not perfect, since the zonefile might have a different authority record than the actual authority of a domain name, but it is a best effort. After sorting the authorities based on the number of domains they serve, we shuffle the largest ones among the long-tail of small ones, so that the requests that big authorities will receive shall be spread out throughout the day, as evenly as possible. That way, we are able to avoid DDoS-ing servers on the Internet and eliminate abusive behavior from Thales 2.0.

Third, as mentioned earlier, a large amount of queries will be sent to authorities around the world. With tens of workers submitting resolution requests, we have a large amount of traffic coming back to our systems. Like DNS Amplification Attacks [110], we send

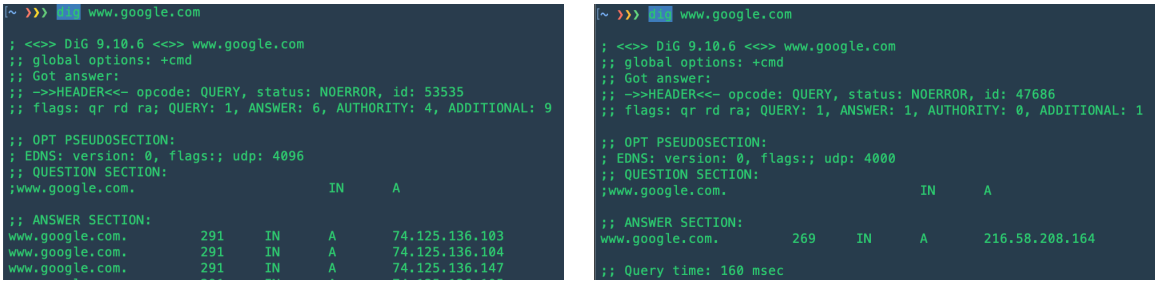


Figure 4.10: Resolution response for `www.google.com` from the US (on the left) and Greece (on the right hand side).

a relatively small amount of data to authorities (a DNS question), but receive a much larger amount of data. A DNS response, is at least as large as the question (almost equal in case of an `RCODE`$\neq 0$), but can grow significantly with multiple responses (up to 13), the `AUTHORITY` and glue (`ADDITIONAL`) sections. Therefore, collecting such vast amount of data, processing it, and storing it in a usable format can be challenging. Our initial decision to use network packet captures (PCAPs) to aggregate data at a single point and then process those PCAPs, required a large processing infrastructure that could handle approximately two terabytes worth of data on a daily basis. Dedicating so many resources meant an increased cost. At the same time, since the PCAP collection was happening at a single place, it was very hard to incorporate higher distribution of worker nodes; each node would need to collect its own data and then push it to the central processing place. Moving away from this architecture and introducing Kafka, allows us to scale horizontally, even in remote locations. Workers can now trivially subscribe to the RabbitMQ queue and push their collected data to the Kafka cluster. From that point on, we can process it in Hadoop-native formats using Spark on our Hadoop cluster.

Fourth, the difficulty of adding remote nodes mentioned earlier, limited the locations from which queries could be submitted from. That is, both network and geographic locations. DNS is using several techniques to load balance and geographically distribute load to different places around the world. For example, when a stub resolves www.google[.]com from the US, it will receive an IP address in the US, usually geographically (or network-

based) close to the its location. On the other hand, someone resolving the same domain from Greece, will receive a completely different response. Figure 4.10 shows an example of the response from Georgia Tech (on the left) and University of the Aegean in Greece (on the right). As we can see, the response is very dissimilar. However, Thales and the way data aggregation was taking place, was very hard to be deployed into different networks and locations in order to identify such differences. Thales 2.0 supports transparent deployment, which can allow for geographically distributed collection nodes, which can all contribute to the same data lake, and hence give us the ability to create a "looking glass" on the Internet for DNS.

After operating the Active DNS collection system for almost five years, we identified issues and potential improvements that were incorporated into Thales 2.0. The latest version of the system can still be extended and further grow through our past experiences and new requirements. As the Internet evolves, so does infrastructure and DNS. The Active DNS dataset is very important in security and network research, hence we plan to keep upgrading and refining Thales 2.0 to enable such research.

4.7 Active and Passive DNS Applications

Passive DNS data has been used by the security community for almost two decades to solve network security problems. The Active DNS dataset, a novel dataset that we introduced approximately five years ago, has been an integral part of research and operational applications. Passive and Active DNS datasets provide significant advantages to researchers and security professionals. Each dataset has its own advantages and disadvantages, but they also complement each other. In this section we discuss the ways Passive DNS data is being used, how Active DNS data has (and can) be used, and what advantages the combination of the two datasets provide.

Before we go into details about each individual dataset, we should point out five important differences among the datasets which will serve as our guide in this analysis. First, and

most importantly, Passive DNS data usually includes client information. This is the *destination IP address* for a DNS response packet. Second, Passive DNS includes significantly more child labels than Active DNS, since it is traffic generated by devices connected to the network that are often connecting to complex infrastructure on the Internet. Third, because devices generate the resolution requests, the DNS traffic in Passive DNS data is also organic network traffic, meaning that it is not systematically generated, but it is human, or arbitrary application, driven. Fourth, Active DNS, as discussed earlier, provides a much higher breadth of data than Passive DNS, since it resolves every domain in the most popular TLDs. Lastly, Active DNS data is generated by queries Thales 2.0 performs by itself, hence, it is not organic and it is instrumented by the system operator. Figure 4.11 depicts these differences and provides an overview of the ways Passive and Active DNS data can be used.

4.7.1 Passive DNS

Passive DNS data has several important advantages and is used in many different applications. This section discusses how Passive DNS has been used and what applications can leverage the data, based on the three differences over Active DNS mentioned earlier.

Clients

Passive DNS data often comes with client IP addresses that have performed a resolution request. This is a very important attribute in several different applications. First, client information is very useful in domain name reputation systems. There are several applications, like Notos [16], that take the number of different clients, the number of queries per client, time distribution per client, and other features, into consideration when they attribute a reputation score to a domain name.

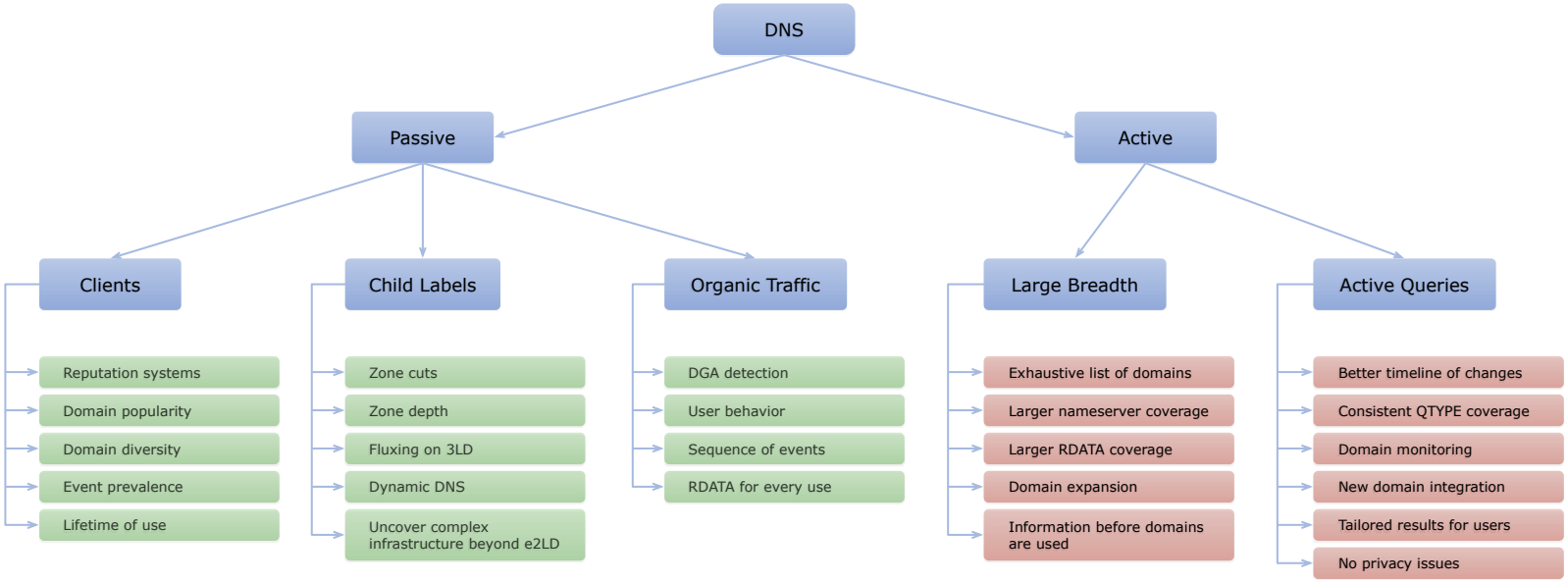


Figure 4.11: Differences between Passive and Active DNS data and the advantages each dataset provides.

Second, client data can provide domain name popularity information. Many requests from many different clients can hint towards a more popular domain name than one that is being heavily resolved solely by one client. Such measurements can also yield information regarding (third) prevalence of a domain name and (fourth) diversity, both geographic and network. Prevalence can help in volumetric statistics of cases like, for instance, malware installation, botnet population, victims of attacks, etc. On the other hand, the geographic and network diversity of domain names can provide information regarding the differences in responses an authority will provide based on the geographic location of a client for load-balancing needs, or the network location, using technologies like the EDNS-Client-Subnet (RFC 7871 [111]).

Finally, Passive DNS data can provide information regarding the time period a domain name has been used for. Since the domain resolutions are initiated by devices in the network or humans using network connected devices, when a domain is resolved successfully, we can assume that most likely the domain is also in use. That contradicts Active DNS for example, which will resolve any domain that is resolved just because it is registered. Hence, Passive DNS can provide *lifetime of use* information, whereas Active DNS can provide *lifetime of existence* information for the same domain.

Child Labels

A *child label* is the part of the domain name "under" the registered portion of the domain (e2LD), as mentioned in Section 2.1.1 and Figure 2.1, p. 9. Passive DNS data will mostly include Fully Qualified Domain Names (FQDNs) in the QNAME field of Resource Records (RRs). On the contrary, Active DNS does not contain a variety of child labels, since it mostly resolves domain names found in TLD zonefiles, or domain names that have been registered (the e2LDs).

Child labels can be very useful when analysis of the structure and depth of a zone is performed. For example, a registered domain like `example[.]com` can have a va-

riety of child labels, like foo.example[.]com and bar.example[.]com. These in turn can have other child labels, like buz.foo.example[.]com, or even a whole new zone under a child label, like zone.bar.example[.]com. This is referred to as a *zone cut*. A zone cut will have a new Start Of Authority (SOA) record and nameservers for the delegated zone. In this example, there might be ns1/2.example2[.com] which are the nameservers (NS) for anything under the zone.bar.example[.]com zone. In order for someone to realize that this is a zone cut, they would need to explicitly submit a query that asks for the SOA record of zone.bar.example[.]com. If a response exists, then this is a new zone. Passive DNS data could have such information, since systems could arbitrarily be querying for this, as long as they know that the FQDN zone.bar.example[.]com exists. On the other hand, Active DNS, which would rarely know that this FQDN exists, will probably miss this delegation and the entire zone.

Such complex infrastructure under a particular e2LD is very rare in malicious domain names. Intuitively, a defender who wishes to protect themselves against malicious communication with a domain name, e.g., a.zone.bar.example[.]com, could simply block a parent domain name, like bar.example[.]com or the entire e2LD. Hence, adversaries often avoid investing into building these complicated structures, since it is trivial for a defender to block the whole zone. However, Passive DNS data will include this structure which can be used to study how an attack might have been rendered or which part of the infrastructure under a zone was malicious. This knowledge of complex zones can uncover cases where we can experience fluxing (or fast fluxing) at a child label level. Still rare, since blocking the e2LD can also block the entire zone, but not impossible to find.

Finally, dynamic domain names from dynamic DNS providers can be found in Passive DNS data, but not very often in Active DNS. Dynamic DNS providers, like Dyn, easyDNS, No-IP, DNSimple, etc, register domain names under TLDs and then provide customers with the ability to register an arbitrary domain under those zones. For example, a dynamic DNS

provider would register `dynamic[.]com` and then allow customers to register domains like `foo.dynamic[.]com` and `bar.dynamic[.]com`. Active DNS will be able to see that `dynamic[.]com` was registered under the `com` TLD, but will have no information about other domains registered under that zone. Since dynamic DNS providers will rarely share the zonefile for their zones, it is impossible to know every permutation of a registered domain under a dynamic DNS zone. On the other hand, if a dynamic DNS FQDN is in use, it will be visible in Passive DNS data, when someone resolves it.

Organic Traffic

Passive DNS data includes domain names that have been resolved by a device in the network being monitored. Such resolutions can be stimulated either by a user or a system while operating. Domains resolved by users can uncover user behavior and what users are trying to browse or what services they are trying to use. For example, when a domain like `netflix[.]com` is resolved in a network, a user might be trying to browse content on Netflix. If such resolution is then followed by resolutions to the Netflix Content Delivery Network (CDN), then there is a high probability that the user is streaming content from Netflix. Such patterns can allow analysts to infer user behavior and understand how a network is being used.

This sequence of events that allows one to infer that a user is streaming content from a content provider could also be used to infer information about the lifecycle of malicious software. For instance, if a malware binary is performing a particular sequence of resolutions every time it runs, or the first time it is executed, looking for this pattern in Passive DNS data, can yield information about a network being compromised. Resolutions of the same domain names in a particular order and timing could yield information about the applications running in that network. The conditional probability of two different applications exhibiting the same resolution pattern drops significantly as the number of unique events increases. However, such study would require a very precise Passive DNS dataset, with

no packet loss and very granular timing information, which is particularly hard to create in practice.

Moreover, organic traffic in the network can be useful to understand how domain names change over short periods of time. Active DNS data has an interval of approximately 12 hours between two resolutions of the same domain name. However, several attacks can be much shorter lived than 12 hours; for example, a phishing or a spam campaign can last a few hours. If those hours fall between the 12 hour window that Active DNS takes to resolve a domain the second time, it will be impossible to know that a domain has changed the infrastructure it was using. Passive DNS, however, will include this information, as long as a client in the network monitored was a victim of such attack. Because the client will need to resolve the domain in order to visit the phishing page or retrieve the spam email, the resolution will be recorded in Passive DNS and the RDATA of the domain will be stored.

Similarly, Domain Generation Algorithms (DGA) are very hard to find in Active DNS data. DGAs are used by malware to find the Command and Control (C&C) server, by randomly generating domains and attempting to resolve them. One of the domains generated will be registered by the botmaster, who will point it to the IP address of the C&C server. Therefore, when one of the randomly generated domains successfully resolves, the compromised host will attempt to connect to the IP it will retrieve from the resolution process. Active DNS will include the randomly generated domain that was registered, but will not include any other domain that was generated, resolved, but resulted in an `NXDOMAIN`. Passive DNS, on the other hand, can have every single resolution request, including the domains that do not exist. Systems like Pleiades [18] rely on these unsuccessful resolution requests to detect previously unknown malware families that utilize DGAs in a network that may have been compromised.

4.7.2 Active DNS

The Active DNS data includes actively resolved domain names from a variety of different TLDs and data sources, including, but not limited to, blacklist, popular domain lists, and partner contribution. This section discusses how Active DNS data has been used by the security community over the last five years and what advantages it provides over Passive DNS data.

Breadth

The Active DNS datasets has a significantly larger breadth than Passive DNS data. Section 4.5.1, p. 71 went into a detailed analysis of the difference between the Active DNS dataset and Passive DNS data collected at a large University. The larger breadth of data provides several advantages to Active DNS, like an exhaustive list of domain names for every TLD Thales 2.0 has visibility into. Since domains resolved come from zonefiles from popular TLDs, every domain that exists will have been resolved at least one day (even in the event of a deletion). As we can see, Active DNS includes millions of domain names, which cover 90% [112] of generic TLDs (gTLDs) and almost 70% [113] of every domain name registered.

The larger breadth of data in Active DNS provides benefits in the analysis of infrastructure used on the Internet. Because of the larger coverage in domain names and RDATA for those domains, Active DNS includes many more IP addresses and nameservers that are used as part of the Internet's infrastructure. Therefore, systems that depend on larger coverage to taint Internet infrastructure based on prior knowledge, can significantly benefit from Active DNS. For example, the Combosquatting Rating System, discussed in Section 5.6, p. 137, uses clusters of connected components of the IP space to group together domain names that could be used for illicit purposes. Similarly, pivoting through infrastructure, either IPs or nameservers, can expand knowledge we have on particular threats. For instance, if we know that a certain domain name is malicious, we can use Active DNS to get every

other domain immediately related to the malicious domain, when they share the same infrastructure. This would have been impossible in Passive DNS, unless all the domains used on the malicious infrastructure had been resolved by the clients in the network monitored. Hence, if an adversary is using multiple domain names for different targets to reduce exposure, Passive DNS would only record the domains resolved in the network monitored, but would have no visibility into the rest of the infrastructure. On the other hand, Active DNS will be able to uncover as much of the infrastructure as the adversary has registered under 75% of TLDs on the Internet.

Finally, the large volume of domain names and the exhaustive lists under the TLDs Active DNS has visibility into provide timely information about registrations of malicious domains. As we saw in Sections 3.4.1, p. 40 and 4.5.2, p. 85, Active DNS includes domain names several weeks, even months, before they appear in public blacklists or malware network traces. Therefore, researchers can perform post-mortem analysis on threats and the way they manifested themselves in the past, and utilize Active DNS data to better detection systems or build new ones based on the dataset.

Active Queries

The Active DNS dataset is composed of domain name resolutions that have been made by Thales 2.0. Hence, the dataset provides more granularity in the domains that are being queried and a more stable resolution pattern, as we saw in Figure 4.4, p. 76. The consistency of queries and resolutions allows for better completeness in the dataset across time. Passive DNS data only includes domains that have been resolved in the network, which may be prone to gaps when a domain is not resolved for a few days, or even mislead analysts if a domain stops being resolved. Active DNS, however, will provide a more complete view of the Internet, since a domain will be resolved, as long as it exists. At the same time, changes to the RDATA for a particular domain will be recorded every day in the Active DNS dataset, but only upon resolution in the Passive DNS data.

Domains in Active DNS are being resolved daily and every specified QTYPE is queried. Hence, the Active DNS data will have a steady supply of RDATA for every QTYPE, as long as there is an associated response when a query takes place. As we saw in Figure 4.7, p. 83, there are several QTYPEs, like `MX`, `SOA`, and `TXT`, where Active DNS will include from two to four orders of magnitude more data than Passive DNS. Systems and analysis that rely on QTYPEs not very popular in Passive DNS, can significantly benefit from Active DNS, like the work from Portier et al. [114].

Since domains are actively resolved, this allows Active DNS to be tailored to particular problems, as well. For example, it is easy to increase the query rate of specific domains of interest when a malicious campaign is being monitored, by just adding the same domains multiple times in the seed queue. Hence, if an event requires more thorough analysis, it would be possible to approximate queries to a set of domains in a periodic fashion. It would be as easy to include more domains from different partners and expand the list of RRs in the Active DNS data at will. Similarly, if a partner requires data that has to be kept private to them, Active DNS can flag domains and exclude them from the publicly released dataset, but only keep the RRs for the parties they were meant to be used by.

Finally, since the data is queried by an automated system, there are no privacy concerns that could be associated with Passive DNS. Thales 2.0 is resolving domain names that are publicly available on the Internet, or provided under licenses that allow sharing for non-for-profit use. Hence, the scientific community can benefit from the dataset without any restrictions, allowing for research results to be publicly disclosed and experiments to be rerun and independently validated, which is the cornerstone of the scientific process.

4.7.3 Combining Datasets

As mentioned earlier, each of the Passive and Active DNS datasets has its own advantages and disadvantages, but they complement each other. Passive DNS is hard to obtain or collect, and usually expensive to purchase. However, if one has access to Passive DNS, they

can use Active DNS to enhance their studies and get comprehensive and more complete results.

There are several cases where both Passive and Active DNS data has been used with significant contribution. For example, the Combosquatting work, discussed in Chapter 5, p. 108 and [80], showcases results of a five year study of combosquatting domain names, with behavioral characteristics in the Passive DNS data, but coverage and prevalence data from the Active DNS dataset. Similarly, the Mirai work from Antonakakis et al. [27], takes advantage of Active and Passive DNS data to expand on the domain names that pointed to IP addresses known Mirai C&C domains were pointing to. From that point, they utilize Passive DNS to further identify the prevalence of the botnet in the population. Active DNS provided more domain names that could be related to Mirai and Passive DNS allowed to further identify clients that were running the malware.

When overlapping Passive and Active DNS data is available, like in the Mirai study [27], Active DNS can be used to pivot from known malicious indicators and expand to more related domain names. Then, Passive DNS can be utilized to, first, identify potentially false positives (FPs), and, second, further measure the impact of an attack or security problem. The inverse is also possible; identifying an interesting trend in Passive DNS, like a new domain name resolved in a network, then use Active DNS to validate whether that domain name is indeed newly registered, if it has changed ownership, if the domain is pointing to known benign or malicious infrastructure, or simply find other related domains. This *pivoting* approach can be also performed via other DNS datasets. For example, malware network traces, DNS blacklists (DNSBL), Open Source Intelligence (OSINT), and other datasets that include domains or RRs can be used to increase the knowledge an analyst has about resolution requests that take place in their network.

Moreover, Active DNS can complement the visibility coverage Passive DNS has. For example, Passive DNS data can be heavily biased by user behavior in a network. Passive DNS will most often exhibit a diurnal pattern in the number of resolution requests through-

out a day, with a peak during work hours and low at nighttime. Then, weekends can be very harmful, when activity in the network is very limited and usually comes from automated systems, like server updates, or email. On the other hand, Active DNS will sustain a stable data rate across all days of the week. Hence, the *local visibility* that Passive DNS provides in a particular network, can be complemented by the *global visibility* Active DNS can provide for every domain name it has resolved, and keeps resolving irrespective of weekends or holidays. Hence, analysts can rely on Active DNS for features like existence, changes in the infrastructure, detailed data for different QTYPEs, and on Passive DNS for features like resolutions from their local network, time of resolution, client information. Therefore, both datasets can be used by the security community to reason about hypotheses. Active DNS can provide the yardstick to validate results that stem from Passive DNS and Passive DNS can provide further insights into a local network that is being studied.

When it comes to DNS data collection, Active DNS can be a significant asset for more granular analysis of particular infrastructure. Former work, like *zmap* [115], has demonstrated how IPv4 scanning can take place in a scalable and timely manner. Such systems allow for data collection that describes the IP infrastructure in use. Similar to *zmap*, the Active DNS system can be used to target a predefined part of the Internet infrastructure and collect detailed information around the way Infrastructure is used. This information can span from reverse DNS pointer records (i.e., the host names assigned to an IP address) to domain name tracking over small or large periods of time. More specifically, given a set of IP addresses, the Active DNS system can be configured to perform reverse pointer record lookups (`PTR` QTYPE) and horizontally scale to account for the frequency the user requires. With post-processing of the Active DNS dataset, we can identify domain names that point to that IP space and authorities in that IP space that serve domain names. Using that knowledge, we can then update the queue of the domains that are being resolved, increasing the frequency with which we can collect data for infrastructure within the IP space we care about. Hence, we can increase the data collected for part of the Internet we

are more interested into and adjust the data granularity based either on external knowledge (e.g., particular infrastructure known to be used by adversaries), or via a feedback loop from the Active DNS system itself, where we observe changes and further enhance data collection into networks of interest.

Finally, as mentioned earlier, Passive DNS provides a significantly larger depth into DNS zones than Active DNS. This results into a variety of child labels in Passive DNS datasets that are uncommon in Active DNS. At the same time, Passive DNS might also include domains from ccTLDs that may not be part of the TLD zones that Active DNS has visibility into. Combining the two datasets can allow for an increase in the Active DNS seed, which in turn can result in much more data actively collected. However, in both cases, potential privacy implications might rise. Several organizations are employing DNS policies like DNS Response Policy Zones (RPZ) [116] and Split-horizon DNS [117], under which a local recursive and authority will provide different answers based on the domain queried and the Internet location of the client submitting the query. For instance, a recursive might reply with a different IP address to clients from within the network it serves for domains that network administrators would like to block. In that case, trying to recreate such RRs in Active DNS might lead to a much different view of the Internet from the two vantage points, rendering correlation misleading. Similarly, a local recursive might be configured to reply for TLDs that do not exist on the Internet, but are used internally in a network. Such examples include the very common `.local` TLD, which is used to resolve IPs for local devices in the network. Domains under that zone might include host names and device identifiers, that if used in the Active DNS seed, could end up in Passive DNS data collected at different vantage points, like the root servers or Autonomous Systems outside of our control. This issue can be generalized to arbitrary child labels found in Passive DNS datasets in general. Even though DNS data is expected to be public, the structure of a zone can be kept private if the domain owner does not wish to share it. However, once an FQDN from that zone is resolved, the child labels are available in Passive DNS

and could be used for network enumeration and reconnaissance. As we mentioned earlier, Passive DNS usually comes with certain legal obligations to protect such leaks. Therefore, extracting zone information and publicly disclosing it in a dataset like the Active DNS one, might lead to privacy concerns, that could possibly fall into the realm of the General Data Protection Regulation (GDPR) [118].

CHAPTER 5

COMBOSQUATTING DOMAIN NAME THREATS

5.1 Introduction

The Domain Name System (DNS) [13, 14], is a distributed hierarchical database that acts as the Internet's phone book. DNS's main goal is the translation of human readable domains to IP addresses. The reliability and agility that DNS offers has been fundamental to the effort to scale information and business across the Internet. Thus, it is not surprising that miscreants heavily rely on DNS to scale their abusive operations.

In fact, domain squatting is a very common tactic used to facilitate abuse by registering domains that are confusingly similar [119] to those belonging to large Internet brands. Past work has thoroughly investigated typosquatting (domain squatting via typographical errors) [51, 50, 120, 121, 122, 123], bit squatting (domain squatting via accidental bit flips) [124, 53], homograph-based squatting (domains that abuse characters from different character sets) [125, 126], and homophone-based squatting (domains that abuse the pronunciation similarity of different words) [54].

A type of domain squatting that has yet to be extensively studied is that of "combosquatting." Combosquatting refers to the combination of a recognizable brand name with other keywords (e.g., paypal-members[.]com and facebookfriends[.]com). While some existing research uses other terms to describe combosquatting domains (i.e.,, "cousin domains" [49]), this work only studies combosquatting in the context of phishing abuse, failing to capture the full spectrum of potential abuse. Thus, even though the general concept of constructing these types of malicious domains is part of the collective consciousness of security researchers, the community lacks a large-scale, empirical study on combosquatting and how it may be abused. Therefore, the security community has little insight into

which trademarks domain squatters commonly abuse, how well existing blacklists capture such abuse, and which types of abuse combosquatting is used for.

In this chapter, we present the first large-scale, longitudinal study of combosquatting abuse to empirically measure its impact. By combining more than 468 billion DNS records from both active and passive DNS datasets, which span almost six years, we identify 2.7 million combosquatting domains that target 268 of the most popular trademarks in the US, and we find that combosquatting domains are 100 times more prevalent than typosquatting domains—despite the fact that combosquatting has been less studied. Our study also makes several key observations that help better characterize how combosquatting is used for abuse.

5.1.1 Contributions

First, we study the *lexical characteristics* of combosquatting domains. We observe that combosquatting lacks generative models and find that, while combosquatting domains vary in overall length, 50% add at most eight additional characters to the original trademark being abused. Furthermore, 40% of combosquatting domains are constructed by adding a single token (Section 5.4.2) to the original trademark. Thus, while the pool of potential combosquatting domains is very large, we find that many instances of combosquatting try and limit the overall length of the combosquatting domain. Additionally, we find that combosquatting domains tend to prefer words that are closely related to the underlying business category of the trademark—resulting in combinations that are more targeted than random.

Second, we analyze the *temporal properties* of combosquatting domains and, surprisingly, we see that almost 60% of the abusive combosquatting domains can be found in our datasets for more than 1,000 days—suggesting that these abusive domains can often go unremediated. When combosquatting domains *do* become known to the security community, it is often significantly after the threat was seen in the wild. For example, 20% of the abusive combosquatting domains appear on a public blacklist almost 100 days after

we observe initial resolutions in our DNS datasets, and this number goes up to 30% for combosquatting domains observed in malware feeds. To make matters worse, we observe a growing number of queries to combosquatting domains year over year, which is in stark contrast to better known squatting techniques like typosquatting. Thus, combosquatting appears to be an increasingly effective technique used by Internet miscreants.

Third, we discover and analyze numerous instances of combosquatting abuse in the real world. Through a substantial crawling and manual labeling effort, we discover that combosquatting domains are used to perform many different types of abuse that include phishing, social engineering, affiliate abuse, and trademark abuse (i.e., capitalizing on the popularity of trademarks to sell their own products and services). By analyzing publicly available threat reports, we also identified 65 combosquatting domains that were used by Advanced Persistent Threat (APT) campaigns. These findings highlight the wide reaching impact of combosquatting abuse. Finally, we manually analyzed various techniques attackers used to drop malware and counter detection—leading to some interesting discoveries surrounding the use of redirection chains and cookies.

In summary, combosquatting is a type of domain squatting that has yet to be extensively studied by the research community. We provide the first large-scale, empirical study to better understand how attackers use combosquatting to perform a variety of abusive behaviors. Our study examines the lexical characteristics, temporal behavior, and real world abuse of combosquatting domains. We find that not only does combosquatting abuse often appear to go unremediated, but its popularity also appears to be on the rise.

5.2 Squatting Background

In this section, we define combosquatting and discuss how it differs from other types of DNS squatting. Additionally, we discuss how combosquatting is used to facilitate many different types of abuse. For example, Internet miscreants use combosquatting to perform social engineering, drive-by-download attacks, malware communication, and Search En-

gine Optimization (SEO) monetization. Thus, even though combosquatting has not been extensively studied, it has far reaching implications.

5.2.1 DNS Squatting & Combosquatting

Combosquatting refers to the attempt of "borrowing" a domain name's reputation (or brand name) characteristics by integrating a brand domain with other characters or words. Combosquatting differs from other forms of domain name squatting, like typosquatting and bitsquatting [52], in two fundamental ways: first, combosquatting does not involve the spelling deviation from the original trademark and second, it requires the original domain to be **intact** within a set of other characters. In this paper, we consider a domain name being combosquatting based on the following definition.

Given the effective second level domain name (e2LD) of a legitimate trademark, a domain is considered combosquatting if the following two conditions are met: (1) The domain contains the trademark. (2) The domain cannot result by applying the five typosquatting models of Wang et al. [50].

For example, lets consider the trademark *Example*, such that it is served by the domain name example[.]com and the e2LD of which is *example*. Combosquatting domain names, based on this e2LD, could include any combination of valid characters in the Domain Name System, whether they are prepended or appended to the e2LD. For instance, secure-example[.]com, myexample[.]com, another-coolexample-here[.]com are cases of

Table 5.1: Examples of the different types of domain name squatting for the youtube[.]com domain name.

Domain Name	Squatting Type
youtube[.]com	Original Domain
youtubee[.]com	Typosquatting [123]
yewtube[.]com	Homophone-Based Squatting [54]
youtubg[.]com	Bitsquatting [52]
Y0UTUBE[.]com	Homograph-Based Squatting [125]
youtube-login[.]com	**Combosquatting**

combosquatting. However, wwwexample[.]com and examplee[.]com are not, since they violate the second clause mentioned earlier. Table 5.1 shows examples of the different squatting attacks against the youtube[.]com domain name.

5.2.2 Combosquatting Abuse

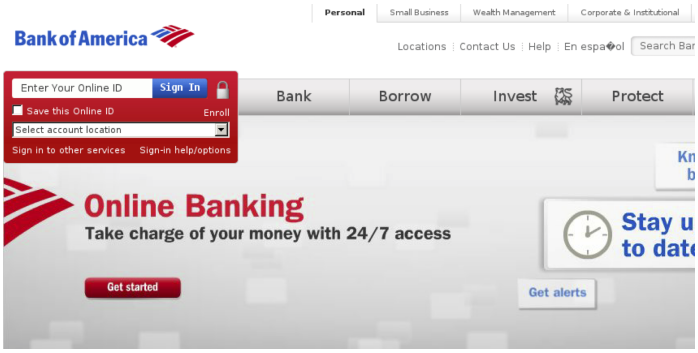
In this section, we discuss the most common types of combosquatting abuse. Despite common beliefs, combosquatting domains are not only used for trademark infringement but are also regularly used in a wide variety of abusive activities—including drive-by downloads, malware command-and-control, SEO, and phishing. We should note that all cases mentioned next were reported to the registrars and law enforcement for remediation.

Phishing

In generic phishing attacks, where obtaining the user's credentials is the final goal of the adversary, the attacker would likely register combosquatting domains close to the targeted organization. For example, in Figure 5.1a we can see one of those phishing campaigns against Bank of America (BoA) users that employees the bankofamerica-com-login-sys-update-online[.]com domain. It is worth noting that the phishing page that was hosted on this combosquatting domain was nearly identical to the actual BoA website. We argue that this visual similarity, when coupled with the bank's brand name clearly embedded in the combosquatting domain, makes it highly unlikely that everyday users of the web would be able to detect this website as phishing.

Malware

Delivery of malware and drive-by attacks is another interesting case of combosquatting abuse. For example, a combosquatting domain can be used to redirect victims to a page showing fake warnings to lure them into downloading malicious software. Figure 5.1b shows the domain airbnbforbeginners[.]com being used to lure new AirBnB users. Once



(a)

(b)

(c)

Figure 5.1: Examples of combosquatting abuse. (a) A typical phishing campaign against Bank of America using the domain bankofamerica-com-login-sys-update-online[.]com. (b) The airbnbforbeginners[.]com domain uses the AirBnB brand to lure users and drop a malware obfuscated as a Flash Update. (c) An example of trademark abuse against Victoria's Secret using the domain name victoriassecretoutlet[.]org.

users land on the page, a Flash update request is shown to the end user in what looks like a Windows dialogue prompt. Thus, the attack attempts to infect the user by using alerts that suggest Flash Player is outdated and then entice the user to download a malicious update.

In Table 5.2, we can see malware related domain names that were used as Command and Control (C&C) points for botnets created using popular malware kits (e.g., Zeus). While it is hard to know for sure why attackers decide to use domains that contain popular trademarks, such domains could evade manual analysis of malware communications. The use of combosquatting domain names is not limited to common malware families, like the ones in Table 5.2. As we will see in Section 5.3.2, using public reports around targeted attacks and Advance Persistent Threats (APTs), we identified more than 60 APT C&C domains that utilize combosquatting, abusing up to 12 different popular brand names.

Monetization

Next to malicious activities mentioned earlier, combosquatting domains have been heavily exploited in trademark infringement and Search Engine Optimization (SEO). In this monetization category, the combosquatting domains often advertise services similar or related to the original services and products offered by the trademarks being abused. A real world example of such a trademark infringing domain is presented in Figure 5.1c in which the domain name victoriassecretoutlet[.]org abuses the Victoria's Secret trademark to offer likely

Table 5.2: Examples of combosquatting domains used by malware as Command and Control (C&C) points.

Domain Name	Trademark	Abuse Type
adobejam[.]in	Adobe	Artro C&C
norton360america[.]biz	Norton	Betabot Botnet
googlesale[.]net	Google	Etumbot
indexstatyahoo[.]com	Yahoo	Phoenix Kit
pnbcnews[.]ru	NBC News	Pkybot Botnet
wordpress-cdn[.]org	WordPress	Pkybot Botnet
youtubeee[.]ru	YouTube	Zeus Botnet
google-search[.]ru	Google	Zeus Botnet

counterfeit products at a lower price.

5.3 Measurement Methodology

Measuring the extent of the combosquatting problem is particularly hard because of the almost unlimited pool of potential domains. However, given the definition of combosquatting in Section 5.2.1, we provide a methodical way to identify combosquatting domains using various datasets. Additionally, we discuss our rationale for selecting trademarks that are most likely to be abused, the type of datasets we use throughout our study, and introduce the necessary notation utilized from this point on.

5.3.1 Trademark Selection

While all trademarks could be the subject of combosquatting abuse, it is arguably not in the best interest of an adversary to use a less known brand for abuse. In our hypothesis we assume that the adversary would include the trademark name in the effective second level domain (e2LD) as a way to lure victims into clicking and interacting with the combosquatting domain and site.

To that extent, we first need to identify the set of popular domains that are used by major brands (likely to be abused by adversaries). To assemble this list of domains, we extracted the top 500 domain names in the United States (US) from Alexa [57]. Our decision to use only the US-centric popular Alexa domains is due to the underlying datasets we will use for our long-term study (which are mostly US-centric), as we will see in the following section.

Now, even with the top 500 Alexa list, not all domains are appropriate candidates for our combosquatting analysis. This is because (1) there are several brands that employ common words as their brand name and (2) there are several domains and trademarks that are too short to be considered for combosquatting. Table 5.3 shows a list of trademarks that were ignored in the Alexa Top 500 due to the previous considerations.

We manually inspected all 500 top Alexa domains to exclude domains that fall into the

Table 5.3: Trademark examples that have been excluded from our study.

Trademark	Domain	Potential Squat
Apple	apple.com	applejuice[.]com
AT&T	att.com	attorney[.]com, attack[.]com
Bing	bing.com	plumbing[.]com, tubing[.]com
citi (bank)	citi.com	cities[.]com, citizen[.]com
IKEA	ikea.com	bikeandride[.]com
Cisco	cisco.com	sanfrancisco[.]com

Table 5.4: Summary of the raw datasets used in this study.

Dataset Type	Size	Records	Time Period	Notation
Passive DNS	18.1T	13.1×10^9	2011–01–01 to 2015–10–14	$PDNS$
Active DNS	30.5T	455×10^9	2015–10–05 to 2016–08–19	$ADNS$
Public BLs	26.7G	610×10^6	2012–12–09 to 2016–09–13	PBL
APT Reports	N/A	21,927	2008–10–01 to 2016–11–04	APT
Spamtrap	35M	965,911	2009–07–17 to 2016–09–13	SPA
Malware Traces	34.8G	1.1×10^9	2011–01–01 to 2016–10–22	MAL
Alexa	42.9G	1.3×10^9	2012–12–09 to 2016–09–13	ALE
Certificate Transparency	842G	271×10^6	2013–03–25 to 2017–04–13	$CERT$

two aforementioned categories. The remaining set contains 246 domains that we will consider in our combosquatting study. We will refer to this list of domains as *seed* throughout the rest of the paper. The trademarks selected belong to companies that are active in different business categories. Thus, we are able to group them together into 22 categories based on the type of services/products they offer.

We derived this categorization using the Alexa list [57], the TrendMicro [127] website and the DMOZ database [128]. We manually verified the categories and merged any differences between the platforms to create a consistent list. The vast majority of the domains had a stable Alexa rank over time. At the same time, we added seven domains that were a priori chosen in the "Politics" category and 15 for the "Energy" category, following the same process as before. We manually included the energy sector because it is part of the critical infrastructure and the politics because of the US Presidential elections of 2016.

Table 5.5: The combosquatting datasets, and their relational statistical properties. NoT: Number of unique trademarks in a set of domains and NoC: Number of unique business categories in a set of domains. $C_{abuse} = \{C_{mal} \cup C_{pbl} \cup C_{apt} \cup C_{spa}\}$.

	Passive DNS			**Active DNS**			
α	$\alpha \cap CP$	NoT	NoC	$\alpha \cap CA$	NoT	NoV	**e2LDs Count**
CP							2,321,914
CA							1,022,083
C_{mal}	9,283	179	21	6,886	174	21	9,472
C_{pbl}	3,750	135	21	4,787	128	21	5,844
C_{apt}	59	11	8	56	12	8	65
C_{spa}	2,296	126	20	6,400	148	20	6,400
C_{abuse}	14,965	201	21	17,586	200	21	21,173
C_{ale}	45,619	244	22	37,098	244	22	48,197

5.3.2 Datasets

Since our goal is to study combosquatting both in depth and over time, we require a variety of different datasets. Table 5.4 summarizes the raw datasets used in this study, and Table 5.5 lists the most important relationships between them. We provide more detail about each of these datasets below.

Passive DNS: The passive DNS dataset ($PDNS$) consists of DNS traffic collected since 2011, above a recursive DNS server located in the largest Internet Service Provider (ISP) in the US. Specifically, this dataset contains the DNS resource records (RRs) from all successful DNS resolutions observed at the ISP, including their daily lookup volume.

Active DNS: We also utilize an active DNS ($ADNS$) dataset, which we obtain daily from the Active DNS project [85]. Since the duration of this dataset is less than a year, it does not have a complete temporal overlap with our $PDNS$ dataset. While we will use the $PDNS$ and $ADNS$ datasets for most measurement tasks, we will also use a variety of smaller datasets to label and measure abuse in these combosquatting datasets. Again, in Table 5.4 we can see these five different datasets used in this study.

Public Blacklists: We collect historic public blacklisting (PBL) information about domains that have been identified by the security community as abusive and placed in various public lists [59, 60, 61, 62, 63, 64, 65, 129]. These blacklists have been collected from 2012 until 2016 and overlap with our passive and active DNS datasets.

Advanced Persistent Threats: Using public Advanced Persistent Threat (APT) reports [1], we manually extract and verify domain names used in such documented attacks (APT).

Spam Trap: A security company provides us with spam trap [130] data that is labeled using their proprietary detection engine (SPA).

Malware Feeds: The same security company and a university provides us with two feeds of domains from dynamic execution of malware samples since 2011 (MAL).

Alexa List: To eliminate potentially wrong classification of a domain as abusive (false positive) in the aforementioned datasets, we create a "whitelist" based on the Alexa list. We take the domains that appeared in the top 10,000 of the Alexa list for more than 90 consecutive days in the last five years and create a set of domains as indicators of benign activity (ALE).

Certificate Transparency: Google's Certificate Transparency (CT) [131] project provides publicly auditable, append-only logs of certificates with cryptographic properties that can be used to verify the legitimacy of certificates seen in the wild. The official CT website provides a list of known, active logs that can be publicly crawled. We used this list to download all records from those logs up to April 13, 2017. This resulted in a dataset of

[1] `http://tinyurl.com/apt-reports`

approximately 271M certificates.

5.3.3 Linking Datasets

Next, we project the selected trademarks, into the raw datasets presented in Table 5.4, and derive the trademark–specific datasets, which can be seen in Table 5.5. The datasets in Table 5.5 will be used to study the combosquatting problem in depth since 2011. We begin by extracting the Combosquatting Passive (CP) and Combosquatting Active DNS (CA) set of domains, which reflect combosquatting domains containing at least one of the trademarks of interest in the Passive and Active DNS datasets, respectively. The cardinalities of these two sets are of the order of millions of domain names (2.3M for the CP set and 1M for the CA), and all combosquatting domain abuse should be bounded by the size of the two sets.

Following the same process, we identify the combosquatting domains in the PBL, APT, Spamtrap, Malware and Alexa sets, deriving C_{pbl}, C_{apt}, C_{spa}, C_{mal} and C_{ale}, respectively. The cardinalities of these sets can be seen in Table 5.5 where they span from a few domains (C_{apt}) to several tens of thousands of domains (C_{mal} and C_{ale}). Finally, we will define C_{abuse} as the set of domains in all malicious categories of combosquatting domains, namely C_{pbl}, C_{apt}, C_{spa}, and C_{mal}.

5.4 Measuring Combosquatting Domains

In this section we present short and long term measurements revolving around the combosquatting domains in our datasets. We begin by investigating the differences between typosquatting and combosquatting. At the same time we discuss which words attackers choose to combine with popular trademarks more frequently. Then, we study the temporal properties of the domain names in the combosquatting passive and active DNS datasets. This analysis will help us understand how these combosquatting domains evolved since 2011.

In particular, we observe that the number of combosquatting domain names in our pas-

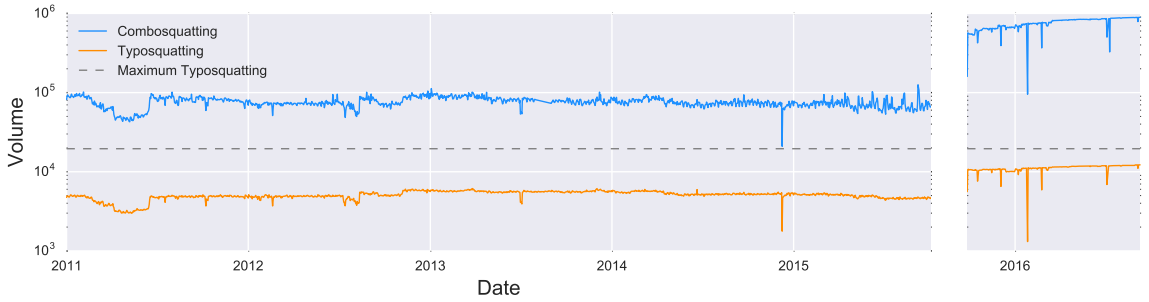


Figure 5.2: Number of active Combosquatting and Typosquatting domain names per day. The left hand side part of the plot depicts the passive DNS period, whereas the right one reflects domains found in the active DNS dataset.

sive and active DNS datasets are steadily increasing; in contrast, the domains in the C_{abuse} set remain stable over time. At the same time, we observer that the security community is lagging behind the detection of malicious combosquatting domains, in many cases up to several months, despite being an obvious target of abuse. Finally, we provide an analysis of the DNS and IP hosting infrastructure that combosquatters tend to employ. The domains in the C_{abuse} set tend to utilize significantly more agile hosting infrastructure, which could be used as a signal to identify abusive combosquatting domains on the rise.

5.4.1 Combosquatting versus Typosquatting

Since typosquatting is, by far, the most researched type of domain squatting, we begin our discussion of combosquatting by comparing it with typosquatting. Figure 5.2 shows the number of active typosquatting and combosquatting domains targeting our evaluated trademarks since 2011. To identify typosquatting domains, we use the five typosquatting models of Wang et al. [50] to generate all possible typosquatting domains and search for those domains in our DNS datasets. The left part of the plot is based on our passive DNS dataset while the right part is based on the active DNS dataset. One can clearly see that, even though combosquatting has evaded the attention of researchers, it is significantly more prevalent than typosquatting, with the number of daily combosquatting domains being almost two orders of magnitude larger than the number of typosquatting domains.

In comparison with other types of domain squatting phenomena such as typosquatting, combosquatting has a unique property in that it lacks a generative model. For all other types of domain squatting, researchers can start with an authoritative domain, and by performing character and bit swaps, they can exhaustively list the possible squatting permutations for a given type of domain squatting. For example, the dotted line in Figure 5.2 indicates the maximum number of typosquatting domains possible when considering the evaluated trademarks and typosquatting models [50]. In combosquatting, however, attackers are free to prefix and postfix a trademark with one or more keywords of their choice, bounded only by the maximum number of characters allowed for any given label by the DNS protocol [30, 31].

Another difference that is closely related to the lack of a generative model, in terms of attack scenarios, has to do with the way attacks are rendered. Typosquatting can be a passive attack for the adversary, who simply must wait until a user accidentally types in a domain. However, combosquatting requires more active involvement from the attacker because, while a user may accidentally type paypa[.]com instead of paypal[.]com, an attacker cannot register paypal-members[.]com and reasonably expect users will accidentally type those eight extra characters. Therefore, miscreants that rely on combosquatting must coerce users (e.g. via spam emails and social networks) to visit combosquatting domains.

To increase the chances that users will interact with their malicious combosquatting domains, attackers can use services like *Let's Encrypt* [132] to both freely and automatically obtain TLS certificates for their domains. In fact, Let's Encrypt has recently come under criticism for choosing to eschew any sort of security checks before giving domain owners a TLS certificate [133]. To quantify the frequency with which attackers obtain certificates for their malicious domains, we searched the 271 million certificates obtained via the Certificate Transparency append-only log (described in Section 5.3.2) and discovered that 691,182 certificates were given to a total of 107,572 fully-qualified combosquatting domains related to our trademarks, since 2013, with 41.5% of the certificates being issued

by Let's Encrypt. In contrast, only 3,011 certificates were issued for typosquatting domains. This finding further confirms the intuition that typosquatting and combosquatting are two distinct phenomena with different threat models and attack strategies.

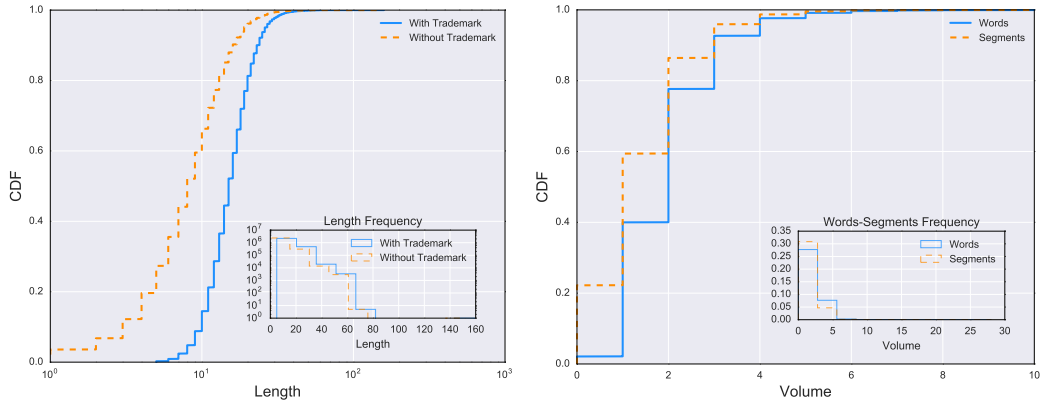
In summary, we argue that existing domain squatting detection systems are not taking combosquatting domains into account (since they cannot generate them) and combosquatting requires its own analysis due to the scale of the problem and the different threat models involved.

5.4.2 Lexical Characteristics

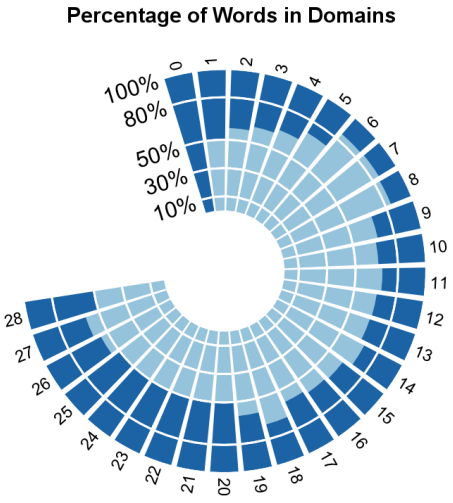
The lack of generative models for combosquatting, makes it hard to proactively create and evaluate domains. Therefore, we utilize the DNS datasets mentioned previously, to identify combosquatting domains and analyze their composition. In particular, we see that adversaries do not usually register lengthy domains and do not use many words when generating the domains. We also find that there are certain words that adversaries favor when generating abusive combosquatting domains. Some words are independent of the trademark's business category, and other words are specific to a single category.

Figure 5.3a shows the Cumulative Distribution Function (CDF) of the length of all identified combosquatting domains. There we can see that even though an attacker can, in principle, construct very long domains, 60% of the identified combosquatting domains were using less than ten characters and 80% of the combosquatting domains were using less than 22 characters (excluding the original squatted trademark). This provides an early indication that the vast majority of the attackers carefully construct combosquatting domain names without attempting to reach the limits afforded to them by the DNS protocol.

To better understand the construction of combosquatting domains, we extract the non-Top Level Domain (non-TLD) part of each domain (e.g. we extract `facebookfriends` from `facebookfriends [.] com`) and use the *word segmentation* algorithm described in [134]. This algorithm takes a string as input and outputs sequences of that string



(a) (b)



(c)

Figure 5.3: Lexical Characteristics of combosquatting domains. (a) Length of the Combosquatting domain names, including and excluding the original trademark. (b) CDF of the number of segments and words. We limit the x-axis of the outer plot for the sake of readability. (c) Number of segments used in combosquatting domain names. For each number of segments the percentage of English words is presented in blue color.

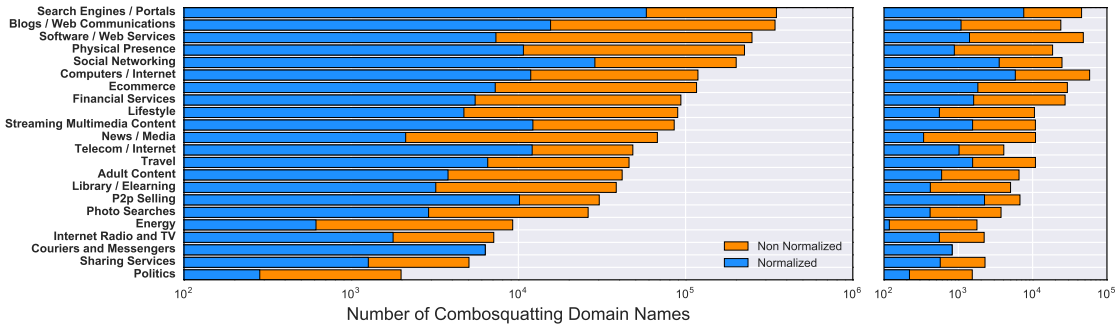


Figure 5.4: Normalized and absolute size of the combosquatting domains in our datasets per business category.

that have a high probability of being standalone tokens, along with a confidence score for the provided tokenization.

We validate the output tokens provided for each combosquatting domain against four dictionaries: (1) the PyEnchant `en_US` Python dictionary [135] to identify English words, (2) the *No Swearing* dictionary [136] to identify swearing-related words, and both (3) the SWOPODS [137] and (4) *No Slang* [138] dictionaries to identify slang words in US English. Tokens that are found in any of these dictionaries are referred to as *words* and, when not found, we simply call them *segments*.

Figure 5.3b depicts a CDF of the number of tokens and number of words that were identified for each domain. We see that almost 80% of the domains have at most two dictionary-words present, and 90% have at most three words. At the same time, we have found a limited number of cases that contain up to 28 words and segments. These results validate our earlier length-based claim that squatters appear to be methodical in their construction of combosquatting domain names. We note that stop words and other short words have not been removed from our datasets because they are frequently used by combosquatting domains.

Figure 5.3c shows the correlation of segments (cyan) and actual words (blue). Every bin in the radial histogram represents the number of tokens identified in each domain. The presented percentage captures the number of actual words versus segments that we were

able to distinguish. As we can see, the middle ranges of token counts (6 to 19) have a lot more segments than words, whereas when the domain consists of fewer tokens, the number of words found in the dictionaries mentioned earlier increases. On average, half of the tokens are words and the other half are segments. This is likely an artifact of the attackers' attempts to register domains that might include typos or several strings close to words, which could be overlooked by the targets, in order to increase their arsenal of combosquatting domains. Consider, for example, the following list of domain names that we identified as combosquatting and all point to a credit card activation campaign.

```
activatemycrbankofamerica[.]com
activatemycrebankofamerica[.]com
activatemycredbankofamerica[.]com
activatemycredibankofamerica[.]com
activatemycreditbankofamerica[.]com
activatemycreditcabankofamerica[.]com
activatemycreditcarbankofamerica[.]com
activatemycreditcardbankofamerica[.]com
```

In terms of the words that attackers combine with abused trademarks, the top twenty words across all trademark categories were: *free, online, code, store, sale, air, best, price, shop, head, home, shoes, work, www, cheap, com, new, buy, max,* and *card*. Since the top twenty words represent all of our 22 categories, they include terms that can be found either in one or multiple trademark categories. For example, the word "free" can be found in 12 of the 22 categories, suggesting that attackers commonly combine the word "free" with popular trademarks associated with paid goods (such as shopping, movies, and TV shows) to lure users into interacting with their websites. Contrastingly, certain words appear in a single category of trademarks, such as "cheap" which is found only in the online shopping category.

Table 5.6 presents the ten most frequent words for each trademark category. We see that many of the popular words closely correlate with the type of trademark being abused, like the words *apple, game* and *phones* being popular in the "Computers/Internet" category and the words *president, vote,* and *elect* being popular in the "Politics" category. The word selection by the adversaries clearly indicates that most registered combosquatting domains have been carefully constructed to match the expected context of each abused trademark. This is a property unique to combosquatting, since any other type of squatting is bounded to the squatted domain name itself. For example, the search space in typosquatting, from which adversaries can choose domain names is bounded to the length of the domain and the characters used, limiting the agility and multiformity of the threat.

Table 5.6: Most frequent words per trademark category.

Category	Most Frequent Words									
Adult Content	free	xxx	porn	sex	gay	live	tube	porno	videos	hot
Blogging	fuck	yeah	love	themes	free	theme	life	blog	best	just
Computers	apple	games	phones	galaxy	phone	office	free	online	support	home
Couriers	office	ground	online	freight	delivery	express	shipping	print	services	service
E-Learning	club	square	school	business	university	health	group	property	online	pilgrim
E-Shop (Auctions)	cars	car	sale	account	south	new	post	posting	san	jobs
E-Shop (Online)	line	store	kindle	online	shop	free	deals	best	lay	card
E-Shop (Physical)	price	sale	store	card	online	prices	home	stores	shop	cheap
Energy	card	cards	online	business	tex	credit	energy	account	chemical	gift
File Sharing	movie	movies	file	free	archive	user	content	login	online	watch
Financial	bank	online	investment	service	account	services	card	worldwide	mortgage	update
Lifestyle	world	land	channel	vacation	games	princess	movie	villa	paris	club
News	news	mike	online	zine	foundation	com	family	new	trust	media
Photography	marketing	photography	photo	buy	time	followers	family	com	photos	best
Politics	president	vote	elect	official	campaign	trump	truth	com	stop	sucks
Radio & TV	free	movies	watch	xxx	movie	chill	account	login	canada	new
Search Engines	plus	mail	search	glass	free	apps	com	play	maps	google
Social Networks	marketing	followers	free	login	buy	account	page	com	business	apps
Software & Web	best	county	new	online	mobile	home	free	sucks	beach	city
Streaming	video	videos	free	download	music	views	converter	best	buy	listen
Telecom	wireless	universal	phone	business	wire	center	online	phones	free	net
Travel	head	island	paris	hotel	garden	inn	hotels	estate	real	beach

5.4.3 Temporal Analysis

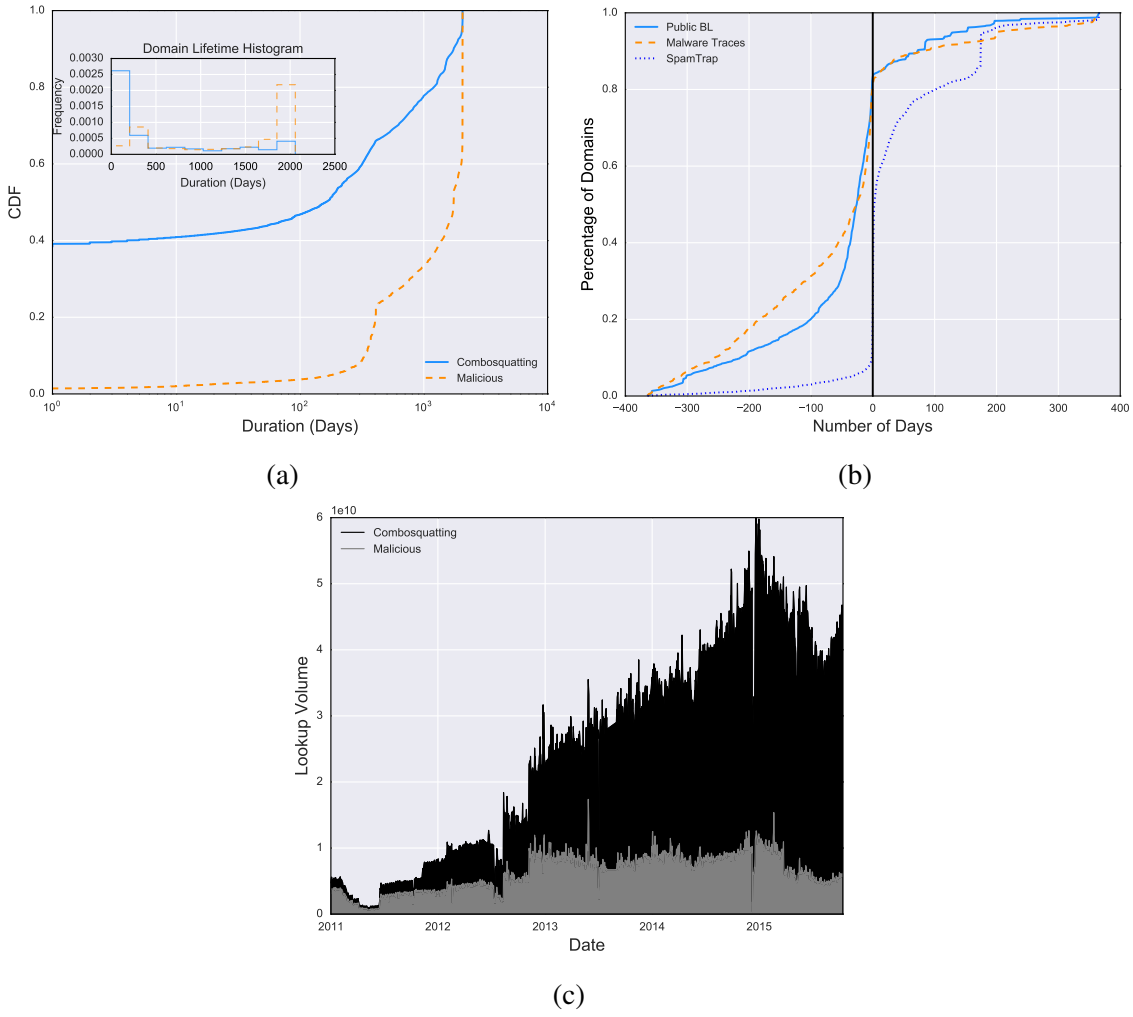


Figure 5.5: Infrastructure characteristics of combosquatting domains. (a) A CDF of the domain name lifetime in the CP set. (b) The difference between the time a combosquatting domain name was first seen in our datasets and the day it first appeared in a Public Blacklist, the Malware Traces dataset, or the security vendor's spam trap. The plot shows the cumulative volume of domains over time, normalized by the maximum number of domains in each dataset. (c) The DNS lookup volume for the domain names in the CP set vs. the malicious (C_{abuse}) domains.

In Section 5.3.1 we presented our process for selecting the trademarks we use in our study, and in Section 5.3.2 we discussed the different datasets we use to measure the phenomenon. Using these trademarks and the dataset notation from Table 5.5, we study the temporal properties of combosquatting domains since 2011. We find that clients are in-

creasingly resolving combosquatting domains and that more than half of all combosquatting domains share a minimum lifespan of at least three months; in contrast, the majority of abusive domains are active for more than a year. We also see that malicious domains appear in the DNS datasets several months before they appear in our abusive dataset and they even make it into the top thousands ranks in the Alexa list.

Figure 5.4 shows the number of combosquatting domain names we were able to place in the passive (left) and active (right) DNS datasets. The orange color represents the total number of combosquatting domain names we are able to identify in our datasets for each of the trademark categories. Blue shows the normalized number based on the number of trademarks that appeared in each category. While most of the combosquatting domain names are in "`Information Technology`" related categories, our dataset is not biased, as the sets CP and CA contain a significant number of domains across all trademarks and business categories.

By focusing our attention on the combosquatting passive DNS set, we can see the days in which a combosquatting domain name is available in our datasets. Figure 5.5a shows the CDF of this lifetime of the domains in the CP set. We measure the lifetime of a combosquatting domain as the number of days between the first and last time we saw it appearing in our passive DNS dataset. Almost 50% of the domain names in the CP set were active for at least 100 days. In the same figure, we can observe the malicious class of combosquatting domain names, which are in the C_{abuse} set (presented earlier in Table 5.5).

Interestingly, Figure 5.5a also shows that the lifetime of abusive combosquatting domains is greater than the entire combosquatting passive DNS set. This makes intuitive sense because a large number of abusive combosquatting domains facilitate malicious network communication for prolonged periods of time.

Figure 5.5b presents how fast the community comes across these combosquatting domains. In the cases of domains from the sets C_{mal} and C_{pbl}, we see that most domains are active several months before they appear in malware traces, or get listed in public black

lists. The only exception is the spam trap that the security vendor is operating, where more than 50% of the domain names in the C_{spa} set appear in passive DNS either a few days before, or on the same day that they appear in the spam trap. One reasonable explanation for this behavior is that it is an artifact of the type of abuse (i.e., spam monetization and social engineering) that these combosquatting domains facilitate.

In order to measure the overall popularity of the domains in the combosquatting passive DNS (CP) dataset over time, in Figure 5.5c we show the DNS lookup volume growth since 2011, according to our PDNS dataset. To put things into perspective, in the same figure, we plot the lookup volume of domains in the C_{abuse} set. It is interesting to observe that while the domains in the CP set have a steady growth over time, the lookup volume of malicious domain names in the set C_{abuse} appears to be nearly uniform. Even though we lack a definite explanation of this behavior, our earlier spam-trap-related results suggest that this almost uniform activity is an artifact of the type of combosquatting abuse (i.e. related to spam and social engineering) that the security industry can reliably detect.

Another interesting observation is related to the Alexa popularity of combosquatting domains. Figure 5.6 shows the distributions of combosquatting domains across the top 1 million Alexa ranks, both for combosquatting domains that are known to be malicious (present in any of our abuse datasets) as well as for all of the remaining combosquatting domains. First, we can observe that, as we move from higher to lower rankings, the concentration of generic combosquatting domains increases. Even so, the overall number of combosquatting domains that are present in the top 1 million Alexa list is limited. In terms of the distribution of malicious combosquatting domains, there we see the presence of malicious domains across all Alexa ranks, which suggests that the existing tools for detecting malicious domains are finding only a small fraction of live attacks, regardless of the overall number of combosquatting activity in any given bin of Alexa ranking. We should note that Figure 5.6 shows aggregate statistics of 20,000 bins in the x-axis. Therefore, the far left domains are cases of combosquatting domains that have made it into any of the top 20,000

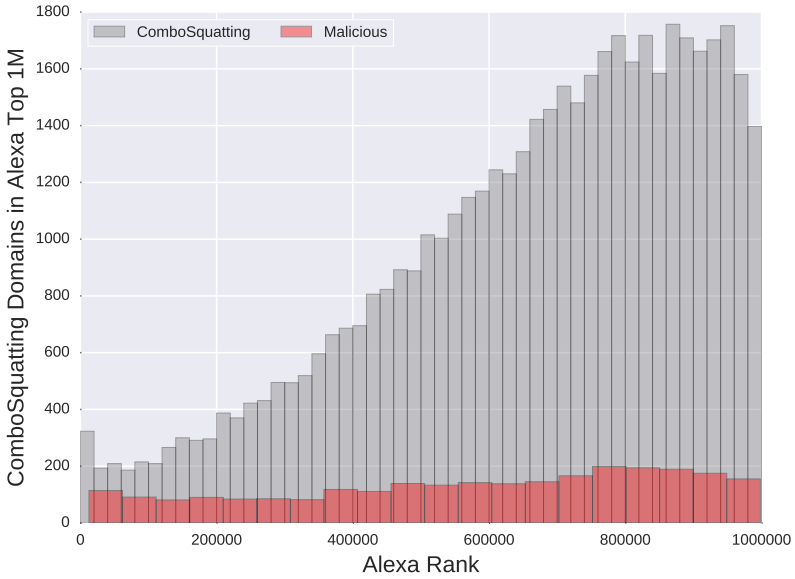


Figure 5.6: Distribution of the Alexa ranks for combosquatting domains since 2011. The plot depicts the mean rank for the domain names over the period of our C_{ale} dataset.

Alexa ranks.

5.4.4 Infrastructure Analysis

So far we have examined how the domains in the combosquatting passive DNS dataset evolved over time. In this section, we turn our attention to the various DNS and IP properties that the domains in the combosquatting passive and active DNS dataset exhibit. We see that the hosting infrastructure of malicious combosquatting domains is concentrated in certain autonomous systems and they are scattered across numerous different CIDRs—which is different from the behavior of combosquatting domains in general.

Figure 5.7a shows the distribution of Classless Inter-Domain Routing (CIDR) networks, Autonomous Systems (AS), and Country Codes (CC) for the hosting facilities of CP and CA combosquatting domains. As expected, generic combosquatting activity is spread across the globe with no obvious concentrations.

We cannot claim the same for the domains in the C_{abuse} set. In Figure 5.7b, we can see a higher concentration of malicious combosquatting domains from the C_{abuse} set in a single

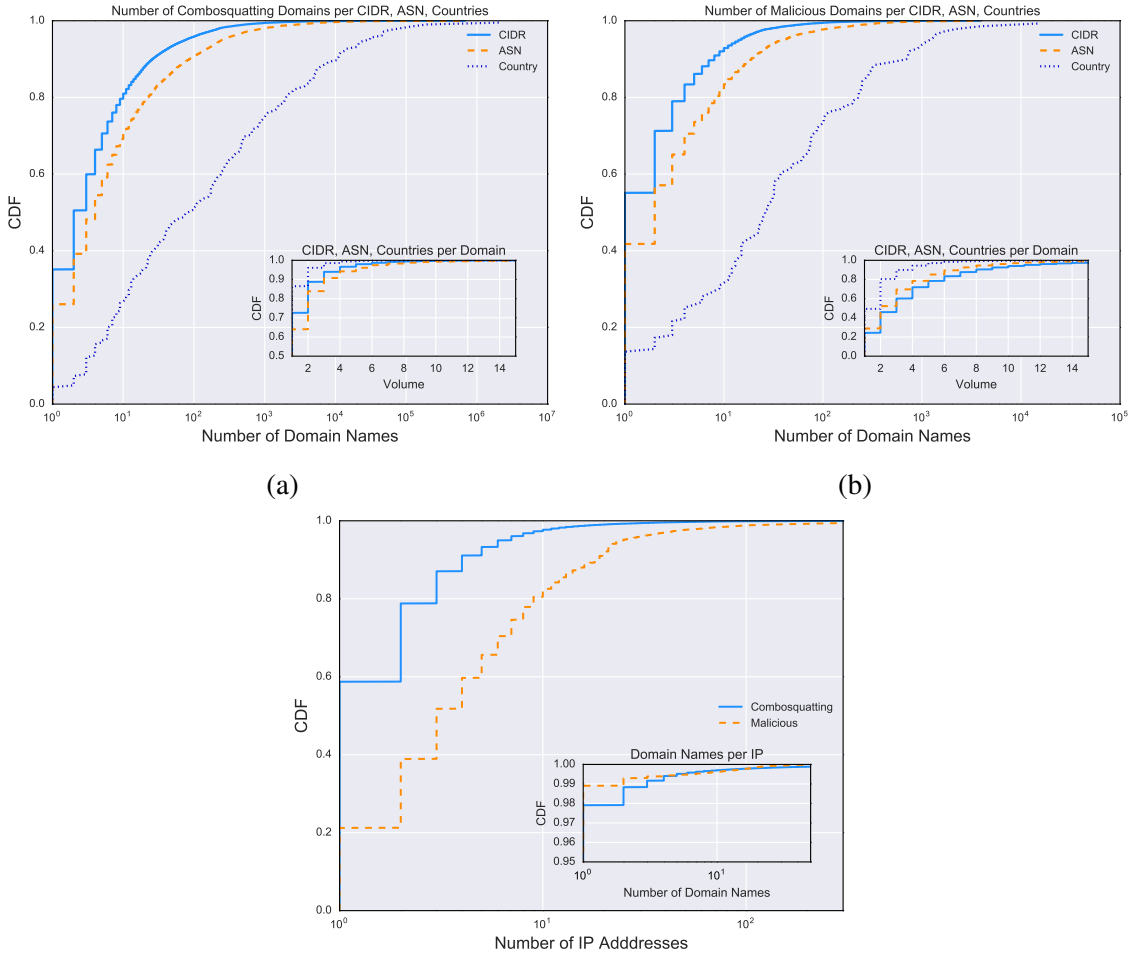


Figure 5.7: Infrastructure distributions for combosquatting Domains. (a) Number of combosquatting domains per CIDR, ASN, and Country for all combosquatting domain names. The inset plot shows the CIDR, ASN, Country Code frequency distribution per combosquatting domain in the CP and CA sets. (b) Number of malicious domains (C_{abuse}) per CIDR, ASN, Countries. The inner plot shows CIDR, ASN, Countries per malicious (C_{abuse}) combosquatting domain. (c) CDFs for the number of IP addresses that domains in the combosquatting (CP and CA) and malicious (C_{abuse}) utilize during their lifetime.

CIDR and AS. That is, almost 58% of the malicious domains are in one CIDR, where only 38% of all combosquatting domains live in a single network. The preference that malicious domains have a single CIDR/AS can be explained in the following two ways. There are few CIDRs and ASes around the world that will permit the long term hosting of malicious domains. At the same time, such malicious combosquatting domains eventually will be remediated, as we saw earlier in this section. This will practically mean that they will be pointed to a DNS sinkhole or a domain parking page.

With this behavior in mind, we tried to better understand both the bipartite graph between the domains in the combosquatting passive and active DNS datasets, and also in the C_{abuse} set. With Figure 5.7c we observe that domains in the set C_{abuse} point to hosts that are spread across more distinct CIDRs than the domains in the CP and CA set. While the rotation on malicious IP infrastructure might not be a new observation, in the reduced space of combosquatting domains, this behavior could be used not only as a way to both track combosquatting domains over time, but also to alert us of potentially new abusive ones.

5.5 Combosquatting in the Wild

So far we have shed light to the combosquatting phenomenon over a period of almost five years. We have shown the complexity of the combosquatting problem by studying its lexical, infrastructure, and temporal properties in Section 5.4. This section focuses on how combosquatting domains are being used in the wild. We study different aspects of combosquatting abuse, at the time of writing, and show how combosquatting can be used for many different types of illicit activities.

We show that combosquatting domains are currently being used for a variety of attacks (e.g. phishing, affiliate abuse, social engineering, trademark abuse). While we study trademarks spread across different business categories, these attacks affect almost every category. We manually analyze a set of combosquatting domains in order to further examine their network behavior and the countermeasures the adversaries take to evade detection.

5.5.1 Exploring & Labeling Combosquatting Domains

In order to understand the current status of combosquatting domains and potential attacks rendered using them, we built an infrastructure of 100 *scriptable* browser instances and used them to crawl 1.3 million combosquatting domains, which were all part of CA (active DNS dataset). The 1.3 million domains were comprised of 1.13 million initial seed domains (note that we have slightly more domains than the ones reported in Table 5.5 since we may crawl multiple subdomains per e2LD). On top of that, we also crawl 200 thousand domains, which included daily registrations of new combosquatting domains and other domains that switched to unknown NS server infrastructure (e.g. non-brand protection companies). Our crawlers were tracking these changes for four weeks and were able to successfully crawl approximately 1.1 million domains.

Due to the sheer size of the collected data and the need of manual verification by human analysts, we approach the dataset we collected through crawling in three sequential steps. First, we scan our entire dataset for evidence of affiliate abuse, i.e., combosquatting domains that redirect users to their intended destination but add an affiliate identifier while doing so. This check will result in the scammer earning a commission from the user's actions [139]. Second, we look in the remainder of the dataset for phishing pages by identifying login forms (from HTML inspection) and focusing on the web pages that are "visually similar" to the legitimate websites. Finally, in order to understand the type of abuse that is neither phishing nor affiliate abuse, we perform a combination of stratified and simple random sampling on our remaining dataset and manually label 8.7 thousand web pages.

All this effort will yield two important points for our study. First, this will help the reader get a sense of how combosquatting is currently used in social engineering and affiliate abuse. Second, we augment the C_{abuse} set of malicious combosquatting domains that escape the threat feeds we used in our study. The next paragraph will provide more details about each step and the discovered abuse.

Table 5.7: Examples of domains used for phishing, as discovered by our crawling infrastructure.

Trademark	#Phishing	Example
Facebook	56	facebook123[.]cf
icloud	48	icloudaccountuser[.]com
Amazon	7	secure5-amazon[.]com
Google	8	drivegoogle[.]ga
PayPal	8	paypal-updates[.]ml
Instagram	7	wvwinstagram[.]com
Baidu	4	baidullhk[.]com

Affiliate abuse First, we scan all pages of our crawled corpus focusing on the ones that, through a series of redirections, navigated our crawlers to the appropriate authoritative domains. By excluding domains that, through their WHOIS records and name servers, we identified as clearly belonging to the legitimate owners of the authoritative domains, we manually investigate the rest of the redirection chains and identify 2,573 unique domains that were, for at least one day, involved in affiliate abuse.

Phishing We scan the HTML code of all the crawled pages that were neither legitimately owned nor abusing affiliate programs, and identify 40,299 unique domains that contain at least one login form. We then proceed to cluster these webpages by their visual appearance using a hamming distance on the hashes produced by a perceptual hashing function, a process which resulted in 7,845 clusters. We then focus on the clusters that contain screenshots that are similar to the look-and-feel of the targeted brands, so as to remove unrelated pages that happen to have login forms. Through this process, we identify 174 domains as conducting phishing attacks. Table 5.7 shows the trademarks that were attacked by four or more combosquatting domains. Even though this number may appear to be small, these were short-lived *live* phishing domains that we discovered in the wild targeting the users of our investigated trademarks.

Table 5.8: Types of combosquatting pages

Unknown	86.6%	Unrelated	11.23%
		Suspicious [1]	88.77%
Malicious	13.39%	Phishing	0.9%
		Social Engineering	13.62%
		Affiliate Abuse	15.56%
		Trademark Abuse	69.9%

[1] Includes under construction, error pages and parking websites.

Other types of abuse Last, we focus on the top two Alexa domains of each of the trademark categories (stratified sampling), resulting in the selection of 221,292 combosquatting domains targeting the selected trademarks. Using perceptual hashing in the same way as we did for the identification of phishing pages, we cluster 351 thousand screenshots of websites (note that many of the 221 thousand combosquatting domains were crawled multiple times due to infrastructure changes that were deemed suspicious) into 50 thousand clusters. The trademark responsible for the largest number of clusters (8.3 thousand) was Amazon which, due to its name, "attracts" thousands of combosquatting websites which are not necessarily related to each other, and thus create clustering singletons. To label the screenshots, we randomly sample 10% of the domains of each affected brand and manually label them, resulting in a manual analysis effort of 8.7 thousand screenshots.

The labeling was performed by the authors where each one chose among the following labels: social engineering (surveys, scams such as tech support scam [140], malicious downloads), trademark abuse (websites capitalizing on the brand of the squatted trademarks), unrelated (seemingly benign and unrelated websites), and error/under construction. Finally, the resulting labels are then used to label the entire clusters in which each sampled screenshot belongs. Table 5.8 shows the overall abuse of the investigated trademarks by consolidating the results of the previous two steps, the manual labeling of the stratified random sample and removing all the authorative domains from the list. Table 5.9 shows the types of abuse for each category of trademarks by focusing on the abuse of its most

Table 5.9: Types of combosquatting abuse for the most popular investigated domain within each trademark category.

Category	Trademark	Phishing	Affiliate Abuse	Social Engineering	Trademark Abuse
Adult Content	pornhub	0%	5.14%	25.73%	69.11%
Blogging	wordpress	0%	0.06%	2.93%	96.96%
Computers	microsoft	0.32%	11.0%	13.68%	74.39%
E-Shop (Online)	amazon	0.36%	61.65%	1.47%	36.50%
Financial	paypal	6.29%	0.78%	55.11%	37.79%
Radio & TV	netflix	2.29%	5.74%	19.54%	72.41%
E-Learning	wikipedia	0%	0%	32.58%	67.14%
Lifestyle	diply	0%	0%	1.6%	98.4%
News	reddit	1.49%	0%	1.49%	97.01%
Couriers	fedex	0%	3.12%	25%	71.87%
E-Shop (C2C)	craigslist	0%	0%	31.10%	68.89%
Photography	pinterest	0%	0%	5.76%	94.23%
E-Shop (Physical)	homedepot	0%	72.5%	2.5%	25%
Search Engines	google	0.32%	3.58%	23.49%	72.32%
File Sharing	dropbox	2.7%	16.21%	51.35%	29.72%
Social Networks	facebook	5.24%	6.18%	18.74%	69.82%
Software & Web	popads	0%	0%	0%	100%
Streaming	youtube	0%	2.02%	14.5%	83.47%
Telecom	xfinity	2.85%	14.28%	11.42%	71.42%
Travel	airbnb	0%	4.04%	1%	94.95%

popular domain (grey cells denote the most popular type of abuse per trademark category). There we see that while trademark abuse is usually the most popular type of abuse, the exact breakdown varies across categories. For example, for both amazon and homedepot, affiliate abuse is the most popular type of abuse, fueled by the fact that these two services offer affiliate programs to their users.

5.6 Combosquatting Rating System

In the previous sections, we have discussed the following; 1) the combosquatting domains are used in a variety of different illicit operations (i.e., Section 5.2.2) and 2) multiple combosquatting domains could be used to support a single malicious campaign (i.e., Section 5.4), and 3) as we saw in Section 5.4, the security community is falling behind the

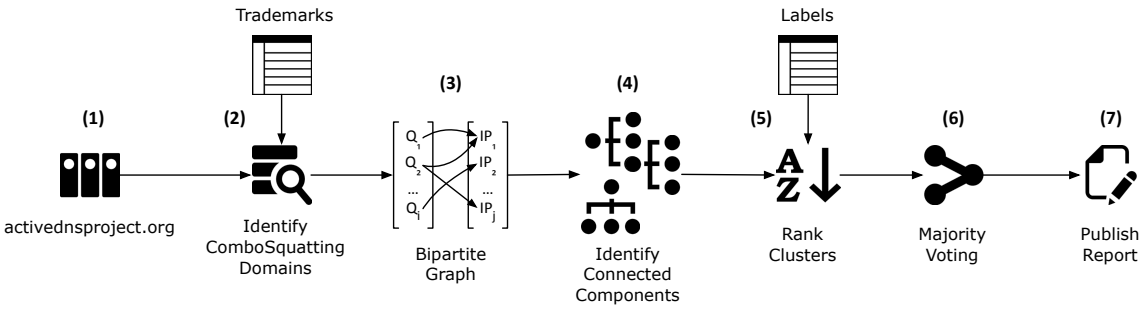


Figure 5.8: Combosquatting Rating (CSR) system. The system receives a daily feed of the public Active DNS data (1) as input to identify combosquatting domains (2), builds associations between them based on a bipartite graph of the domains and the IP address(es) they resolve to (3), identifies clusters of connected components (4), applies known labels and ranks the clusters (5), conducts majority voting on the raked clusters to derive a cluster label (6) and finally publishes the output to the community (7).

detection of such combosquatting domains. We need a system to timely group together and assist in the classification of emerging combosquatting abuse. To achieve that we introduce the *Combosquatting Rating* system (CSR). The goal of the CSR system is to provide to the community a daily, public, feed of ranked combosquatting clusters of domains that are likely to be used for abuse or that have already been involved in illicit activities. With the the permission of the team behind the Active DNS project, we will export our daily system output here: `https://www.activednsproject.org/combosquatting.html`.

Figure 5.8 depicts the overview of CSR. Briefly, the system begins by taking daily DNS datasets from the Active DNS [85] project as input (Step 1). CSR will identify combosquatting domains based on a seed of trademarks provided by the operator (Step 2). The system is able to construct a bipartite graph between every combosquatting domain and the corresponding IP address(es) (in Step 3). In Step 4, CSR will apply a simple connected component discovery algorithm in order to both identify connected components in the association matrix and then build clusters of related combosquatting domains.

The algorithm is a simple modification of the Breadth First Search (BFS). Each day, Algorithm 1 is initialized with the set of combosquatting domains (D) and the set of IP

Algorithm 1 Connected component discovery algorithm.

```
1:  D ← {Combosquatting domains in a day}
2:  IP ← {Resolved IPs of Combosquatting domains}
3:  while IP is not empty do
4:      Initialize a queue
5:      Initialize IP_c and D_c for a connected component
6:      Remove a random ip from IP and add it to queue
7:      while queue is not empty do
8:          Dequeue ip_s from queue
9:          for each d ∈ D that resolved to ip_s do
10:             D_c ← D_c ∪ {d}
11:             D ← D \ {d}
12:             for each ip_t ∈ IP that d resolved to do
13:                 IP_c ← IP_c ∪ {ip_t}
14:                 IP ← IP \ {ip_t}
15:                 Enqueue ip_t to queue
16:             end for
17:         end for
18:     end while
19:     Save the connected component IP_c and D_c
20:     Empty IP_c and D_c
21: end while
```

addresses (IP) they resolved to according to the Active DNS dataset (`Lines 1 and 2`). The algorithm iteratively discovers connected components in the domain resolution graph by using a $queue$ until the set of initial IP addresses IP is empty (`Lines 3 to 21`). `Lines 4 and 5` initialize the necessary data structures for each component. IP_c and D_c will store the IP addresses and combosquatting domains, respectively, for the connected component being discovered in each iteration.

Next, in `Line 6` the algorithm will remove a random IP address from the remaining (not visited yet) set of IPs and add it to the $queue$ — so the process of identifying the current connected component may begin. In the `Lines 7 to 18` the algorithm will perform a breadth first search (BFS) using the $queue$ to store IP addresses needed for BFS traversal. Every time the algorithm dequeues the current IP (ip_s) (`Line 8`), it discovers any not visited yet domains that resolved to ip_s (`Line 9`). Then, these domains are added to the component D_c (`Line 10`). At this point the algorithm also removes these domains from

the not visited yet set D (`Line 11`), such that each domain name will only be visited once. Afterwards, the BFS algorithm identifies more non visited yet IP addresses, saves these IPs to IP_c, removes them from IP and adds them to the $queue$ to be processed later (`Line 12 to 15`).

When the $queue$ is empty, the algorithm has identified one connected component. It saves the IPs and domains of the current connected component to the result set (`Line 19`) and flushes IP_c and D_c to restart for the next component. The aggregated results of Algorithm 1 are clusters of domains that share the same IP infrastructure seen in the Active DNS dataset.

Taking advantage of public information around the domains in each cluster, the system ranks the clusters based on the amount of labeling information that can be attributed to them. Our intuition is that when we know that multiple domains in a cluster appear in a public blacklist, or are known to be bad, then other domains hosted in the same infrastructure might be bad by association, as previous research has shown [16, 41]. At this point, we should note that our goal is not to build a DNS reputation system. Rather, we care to rank clusters of domains in a way that security researchers can quickly identify combosquatting domain names that are likely participating in abuse.

We use three different labels for information that are derived from external sources. We label as *benign* the domains we know are owned by the original owner of the trademark (i.e. `microsoftonline.com`) or have been remediated. Any domain that appears in a blacklist or we cannot be certain it is owned by the original owner, we label it as *suspicious*. In the case where we know that a domain name is not related to either benign or suspicious activity with respect to the trademark it was identified to abuse, we consider it *unrelated*. Such cases can rise from domains that are lexically close to a trademark, but have nothing to do with it. For example, westwitterringbeach.co[.]uk is an odd case that falls into the ComboSquatting definition for Twitter, but has to do with a civil parish in

Southern England [2].

The system utilizes these three classes to rank the clusters based on the amount of known information available to the system (Step 5, in Figure 5.8). That is, for each component, we compute the percentage of labeled combosquatting domains as $p = \frac{L-1}{N}$, where L is the number of labeled domains within a component, and N is the cardinality of the component. This means that p is bounded between zero and one. Because we cannot propagate labels from known domains to unknown domains in singleton components, we offset L by one to give singleton components low ranks. After reversely sorting the connected components based on percentage of labeled domains the system outputs a hierarchical list of clusters and domains that can be then used to identify abusive cases.

Then, the system will conduct a majority voting operation on the ranked clusters using all (at the time) known labels throughout the clusters (Step 6). This process will yield a label for each cluster as being benign, suspicious or unrelated. To break cases of ties between voting, we are being conservative and favor the suspicious class over benign or unrelated. When the tie is between unrelated and benign, we favor unrelated to avoid misleading analysts. Clearly, the operator could provide to the system different (threat specific) lists of labels, and the system will "bubble up" clusters of combosquatting domains that are relevant with this the abuse described in these lists. Finally, the generated reports are being pushed back to the Active DNS project and published to the community so we can increase the situational awareness around the combosquatting abuse (Step 7).

Evasion and CSR We should begin by stating that CSR is not a modeling system. Rather, CSR employs simple graph clustering techniques that are founded on the most basic properties of combosquatting domains; the fact that a domain fits the combosquating definition, and two is hosted in an routable IPv4 IP. If the adversary manipulates the brand name, she will instantly reduce the probability to masquerade the domain as brand related, and certainly not combosquatting. Thus, the only reasonable level of freedom for evasion that the

[2] `http://bit.ly/2f0dYz7`

adversary has is to manipulate the IP(s) the domain points to. This can be done by having a very stable profile (i.e., match one combosquatting domain per IP address). This will force the system to exclude the domain as being a singleton. This is clearly something that CSR cannot deal with, however, this adversarial tactic will exponentially rise the cost of the attack, as the adversary will have to acquire new IP for every domain she uses.

The other way is to point the domain name to enough IPs that will force it to join a "death star" in the CSR domain to IP graph. However, even in this case the adversary will have to point and maintain for a period of time the domain to an IP that actually facilitates the illicit activities. This will create a frequent traversal of the domain name between different clusters with significant different purity scores and domain cardinalities. Clearly, while this behavior could temporarily bypass CSR, a simple modification could capture such adversarial behavior.

5.7 CSR Evaluation and Analysis

In Section 5.6 we introduced CSR system that uses connected component analysis to create clusters of combosquatting domains that share similar infrastructure and compute a rating score for each cluster to identify cases of suspicious domains. We begin by evaluating the connected component clustering in CSR system. Then we provide operational analysis around the number of ranked clusters that CSR produces daily, as the label list changes. Last, we will show how threat researches could use the CSR system output in practice, utilizing a threat console and an example of clustering threat research exercise.

5.7.1 Evaluating the Connected Component Clustering

To evaluate the performance of the connected component clustering of CSR (Figure 5.8, Step 4), we use two metrics: (1) a clustering purity score as external criteria for clustering quality (Section 5.7.1)) and (2) a confusion matrix for the predicted cluster labels verses the actual domain labels (Section 5.7.1).

We use three types of labels to classify combosquatting domains and the derived clusters. A domain can be *suspicious* when there is indication that it has been used in malicious or illicit activity in the past, or when that domain is taking advantage of the trademark it is squatting for reputation purposes (e.g. `google-experts[.]com`). The other two labels are *unrelated*, for domains that have nothing to do with the trademark, and *benign*, for domains owned by the original trademark and not involved in any known malicious related activities. We should note that the labels for malicious and illicit activity were gathered from both the C_{abuse} set but also the domains we manually labeled in the Section 5.5.1. For our evaluation, however, we only select domains that were in our CA datasets between August 8^{th} and August 19^{th}.

Clustering Purity

Cluster purity is defined as *the number of elements that are in agreement with the respective cluster label for the cluster they belong to, divided by the total amount of elements*, according to [141]. In our case, for every domain name in a cluster, we have one of the three aforementioned labels (benign, unrelated and suspicious). The clusters are assigned the label of the majority of the elements they contain. Given cluster C of size $N = |C|$, assuming that B is the set of benign domains, U is the set of unrelated domains and S is the set of suspicious domains, then the label of C, L_C is equal to the label of the largest set: $L_C = L_{\max(|B|,|U|,|S|)}$.

Now, the purity is computed as $\frac{\sum_i \max(|B_i|,|U_i|,|S_i|)}{\sum_i N_i}$. This means that bad clustering will result in purity scores close to 0, whereas good clustering should yield purity scores closer to 1. If every domain is in its own cluster, the purity would be 1. To discount the fact that singletons bias the purity score, we excluded all the singleton clusters before we compute purity.

We used the aforementioned labeled domains to evaluate clustering purity. Table 5.10 shows the purity of our clustering results per day. The average purity is at 0.954195 (or

Table 5.10: Cluster purity per day for our input data. The mean and standard deviation are presented in the last row.

Date	Score	Date	Score
2016-08-08	0.955160	2016-08-09	0.954754
2016-08-10	0.954813	2016-08-11	0.954961
2016-08-12	0.955107	2016-08-13	0.955175
2016-08-14	0.941748	2016-08-15	0.956664
2016-08-16	0.954902	2016-08-17	0.955351
2016-08-18	0.956209	2016-08-19	0.955497
Mean	0.954195	**Standard Dev**	0.003792

95.42%) and the standard deviation is 0.0038 (or 0.38%). It means 95% of domains from non-singleton clusters are in agreement with the majority label. The high purity score indicates that we can cluster related combosquatting domains together very reliably. This also means that clustering makes the system easier to propagate labels from known combosquatting domains to unknown ones. While we showed that CSR is able to achieve good purity rates, we have not shown if the predicted cluster label fits the labels of each domain. In an ideal case, we should have pure clusters with all the domains in the same label.

Confusion Matrix

To measure the confusion between cluster labels and the domain labels, we compose the confusion matrix in Table 5.11. The rows refer to the CSR predicted cluster label (using the previously described majority voting process) and the columns represent the labels for the actual domains in the clusters. Thus, a benign domain name that is accidentally clustered with a majority of suspicious domains, will be assigned the wrong label. The confusion matrix shows us that CSR wrongfully placed 8 benign domains into cluster with predominately suspicious domains. Despite that, most of the domain were placed into a cluster with the same label (as we can see from the diagonal). The unrelated instances do not represent benign or suspicious combosquatting domains and are out of the scope of the system. However, CSR can successfully cluster a large portion of these domains

Table 5.11: Confusion matrix computed on 2016-08-08, between the predicted cluster labels and the actual domain labels in each cluster. For example, we can see that 1,031 suspicious domains ended up in clusters that were predicted as suspicious. However, we had one suspicious domain in a benign cluster and 8 in unrelated clusters.

		Domain Label		
		Unrelated	**Benign**	**Suspicious**
Cluster Label	**Unrelated**	51	0	8
	Benign	1	262	1
	Suspicious	43	8	1031

together successfully. Lastly, there were six cases of a tie for the cluster label assignment, for 14 domains. They only contained an even number of unrelated and suspicious labels, thus the clusters were labeled as suspicious, based on the tie-breaker condition discussed in Section 5.6.

5.7.2 Ranking Cluster Behavioral Analysis

Next, we will discuss how different input lists affect the ranking and majority voting of the CSR system. We will provide real world results over five consecutive days as we operate CSR with four different label lists, namely the C_{pbl}, C_{mal}, C_{spam}, C_{abuse}. In this very typical operational scenario, we need to see how CSR will behave when the operators input different lists for potentially different classes of abuse.

To conduct this experiment we employ the malicious lists already discussed in Table 5.5, namely C_{pbl}, C_{mal}, C_{spam}, C_{abuse} as different seeds to the system. Then we measure the number of labeled vs. unlabeled domains in the top 50 clusters over five consecutive days of operating CSR. Figure 5.9 shows the probability distribution function (PDF) of the percentage of labeled domain names over the size of each cluster (that is, labeled and unlabeled) between August 8 and August 12. The columns represent one of the different lists we use for label input.

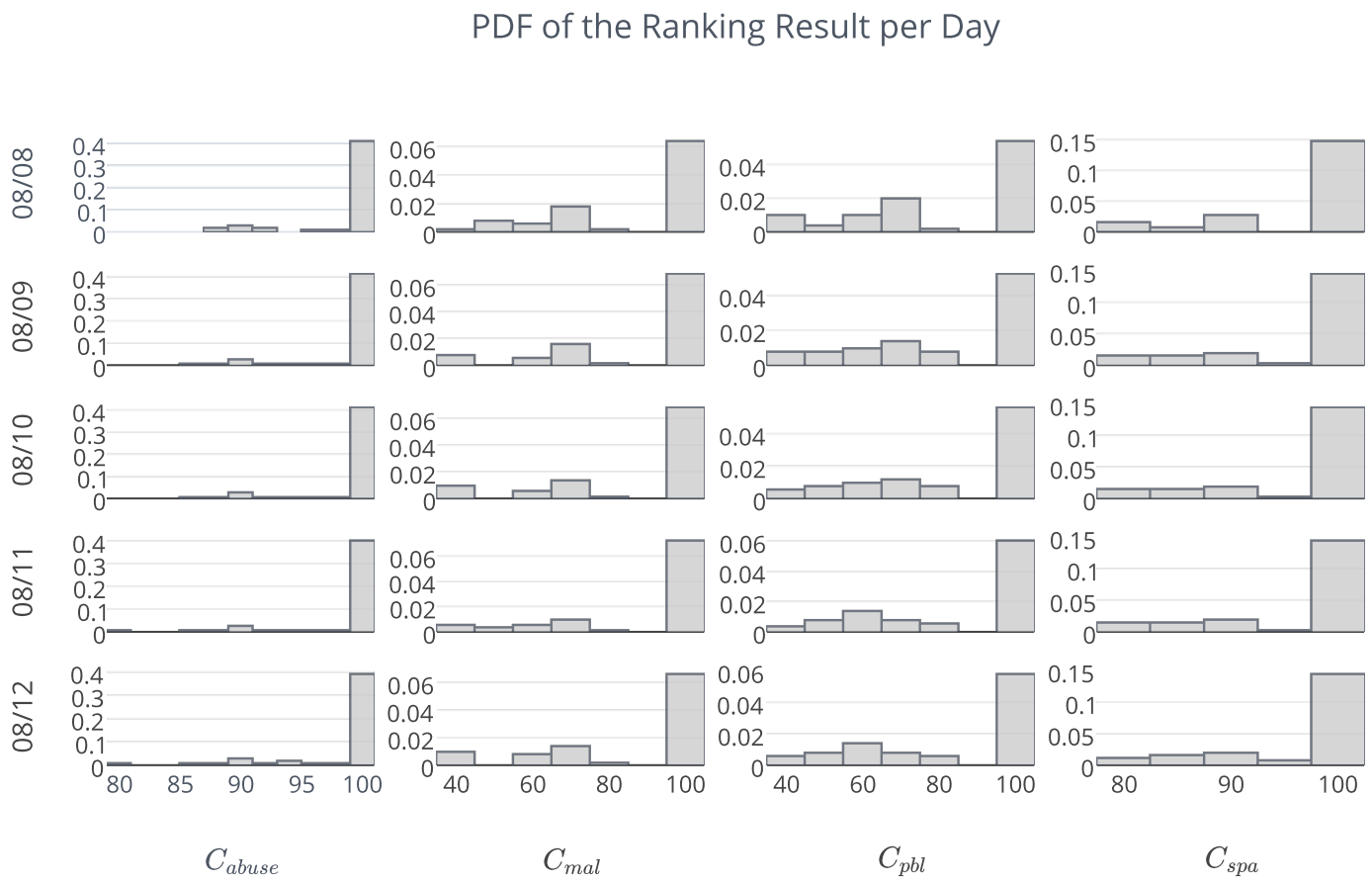


Figure 5.9: CSR ranking results over five days of data. Each column represents a different input labeled dataset. The rows depict days from the past to the future (top to bottom). Recall C_{abuse} is the join of the other three labeled datasets.

We can instantly make three key observations. The first is that, over time the percent of the domains we have a label for in each cluster remains fairly stable per day and per list. This means that the graph of combosquatting domain names we are taking under consideration is fairly static if the observation window is in the matter of days.

The second observation we can make is that, when we combine all the lists together (C_{abuse} set), CSR will yield daily results with the clusters that contain the most labels. We see that up to 40% of the clusters will contain labels for 100% of the domains. Now, when we only use more "noisy" lists (C_{mal} and C_{pbl}) we see that a very small number of clusters (between 4% and 6%) will be fully labeled, and the rest will drastically have less amount of domains that are labeled in them. However, when we use a higher quality list in terms of detection accuracy (like, C_{spa}) we see that the percent of the fully labeled clusters is picking up again (15% of the clusters are fully labeled for C_{spa}).

The third and perhaps most interesting for the users of CSR observation has to do with the amount of clusters that are "mostly" labeled. That is, a threat analyst would like to see clusters with mostly labeled domains, alongside a few unlabeled domains — which she can quickly classify with the same label. In all cases, we see that CSR is able to yield a reasonable amount of clusters (i.e., between 5-15% in the case of C_{pbl}) that have more that 40% of their domains associated with some known type of abuse. This means that CSR will be able to daily produce relevant to the already known labels information for the system's operator.

5.7.3 Using CSR Operationally

In this section we will discuss how the CSR system can be used everyday by researcher. We begin by describing a simple threat console that we implemented in order for the threat researcher to properly interact with the ranked clusters. Then, we will practically show how the clustering results can help guide threat research operations around combosquatting domains.

CSR Threat Console

Visualizing the ranked clustering results can help to quickly understand the structure of clusters and provide insights that support and complement textual reports. With this purpose, we built an interactive visualization tool that represents our clustering results using a circle packing layout. According to the visualization literature [142], the main advantages of this visualization technique are both the good overview of large data sets and the clear representation of groupings and structural relationships. We have already made this visualization public to the community [3] so threat analysts and other security practitioners can easily analyze clusters of domain names related to specific trademarks using our visualization tool. A short demonstrative video that shows how the threat console can be fully utilized is also available at `http://bit.ly/threat-console`.

Particularly, we visually represent clustering results as tangent circles ordered by the rating score of each cluster. Containment within each circle represents a level in the following hierarchy: 1) *cluster*; 2) *trademark*; and 3) *domain name*. For the first level of the hierarchy, *cluster*, the color of each circle represents the three aforementioned labels for ranking clusters, including *benign*, *suspicious*, and *unrelated*. We use blue for clusters ranked as benign, red for clusters ranked as suspicious, and yellow for clusters ranked as unrelated. For unlabeled clusters, we apply gray. For the second level of the hierarchy, *cluster*, we just use containment to represent the different trademarks. Finally, for the third level, besides using color to distinguish domain names' labels, we also use the area of each circle to represent the number of IP addresses pointing out to a specific domain name. Thus, the bigger the circle representing a domain name, the more IP addresses it resolved to. When a specific cluster is clicked, a detailed view is displayed on the right side of the user interface. In the case of clusters, we display the set of related IP addresses to a cluster, name servers, and the list of domain names grouped under such specific cluster. For IP addresses and domain names, we automatically generate links to both Domain Tools and

[3] `https://www.activednsproject.org/combosquatting.html`

VirusTotal in order to assist user queries to WHOIS and AV detection.

Finally, a time-slider is provided in this visualization tool in order to allow understanding the temporal evolution of our clustering results. For large volumes of clustering information, it supports the experience of exploration over time. Figures 5.10 and 5.11 show an overview of the visualization and a specific cluster selected.

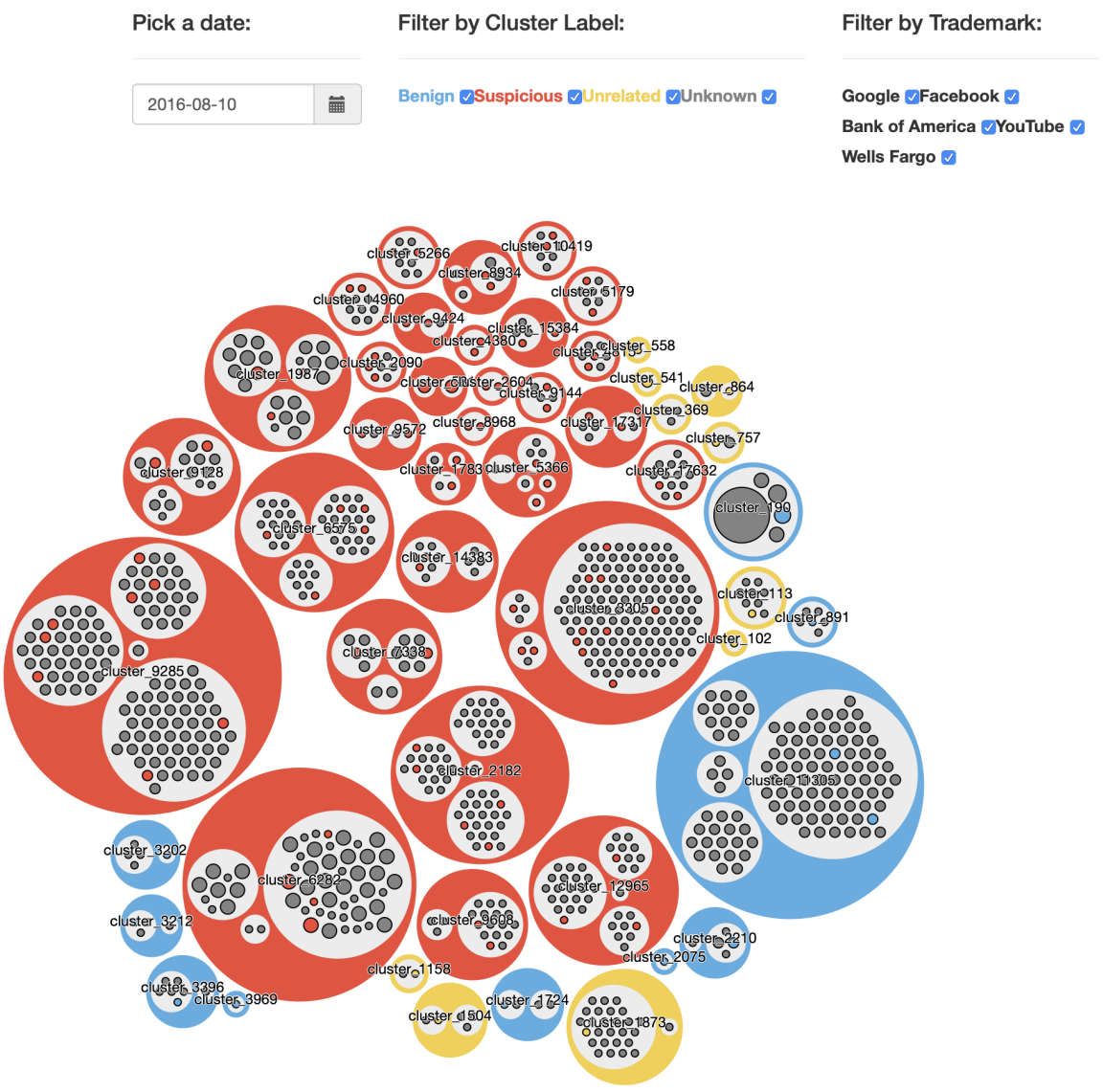


Figure 5.10: Threat console for clustering analysis showing an overview of clustering results for 2016-08-10.

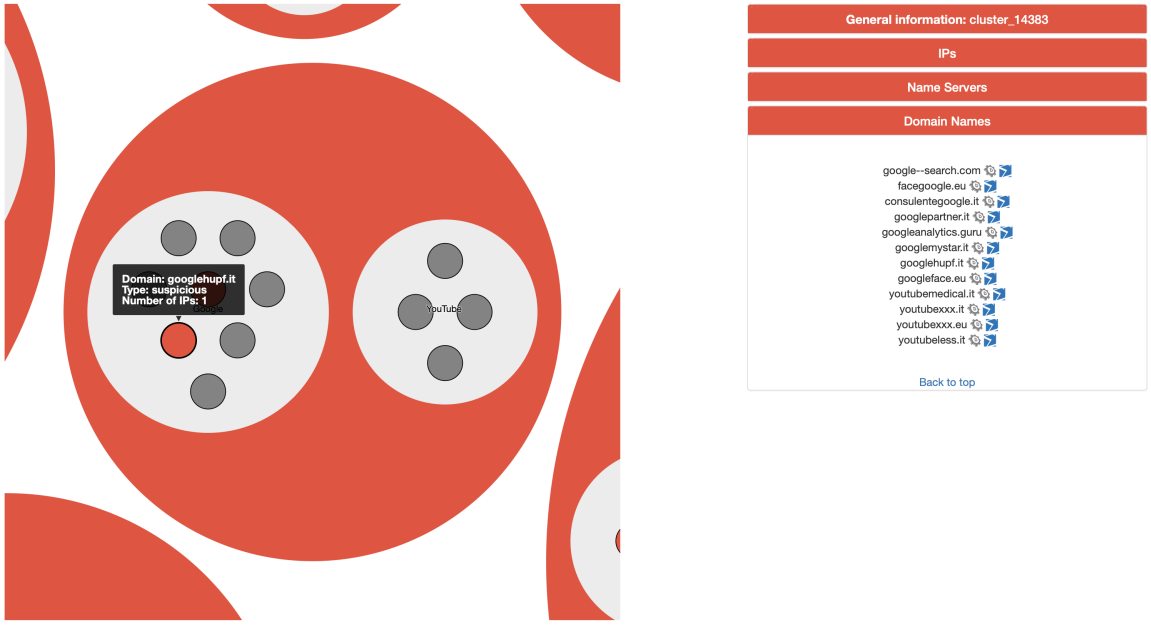


Figure 5.11: Drilling down to Cluster 14383, ranked as suspicious, has been selected, which displays a detailed view of related IPs, name servers, and domain names on the right side of the user interface.

Enabling Threat Research With CSR

Part of our analysis of the clustering results relies on looking into the output of CSR, as it is being released to the community and identify abusive uses of combosquatting domain names. On July 1st, 2016, we identified cluster 92, with a rating score of 0.009 that included nine IP addresses and 606 combosquatting domain names, from 91 different trademarks, including verticals like e-shops with physical presence, blogs, internet services, etc.

This cluster was particularly interesting because of the behavior the domains exhibit when visited. To better understand how adversaries take advantage of combosquatting domains, we setup a headless crawling engine based on the Python *requests* module, that would try to collect Layer 7 (in the OSI stack) information. Our experimental setup was twofold; first we crawled the domains using the default configuration of the module and then we repeated the process specifying a Chrome *User-Agent* in the HTTP headers.

```
1  <html>
2  <head>
3  <title>chevrontexacobusinescard.com</title>
4  <script type="text/javascript">
5  var c = 'f6[...]Sw';
6  var b = String.fromCharCode(61);
7   var a = 'http://park.above.com/jr.php?gz';
8   console.log(a);
9   console.log(b);
10  console.log(c);
11 // window.location.replace(a + b + c);
12 </script>
13 </head>
14 <body bgcolor="#ffffff" text="#000000">
15   <a href="http://www.qfind.net/?__jsfail">
16       Click here to enter
17   </a>.
18 </body>
19 </html>
```

Figure 5.12: The JavaScript redirection performed by some domain names in cluster 92. This example is the result of visiting chevrontexacobusinescard[.]com. Line 5 had a 1,838 characters long string.

Redirection Games On October 30^{th} of 2016, we crawled 505 that successfully resolved and the corresponding service was online. We were able to identify 205 cases where the HTTP server returned an HTTP error code (4xx), whereas 287 replied with `200 OK` status code. Based on our observations, the domains exhibited evasive behavior against our tools, based on factors like HTTP headers, client's IP address and cookies' presence.

Most of the domains in the cluster were associated with a form of redirection, either to a parking page, or to an abuse related website. A set of 114 domains were performing at least one redirection irrespective of the User-Agent HTTP header. When the User-Agent was not set, 28 domains did not redirect and presented a parking page. This set grew to 127 when User-Agent headers were used. Redirection to the parking page was performed via a child label for the same domain name, following the same naming convention: the child label starts with `ww` followed by a number (i.e. starbucksben[.]com redirects to ww1.starbucksben[.]com).

Moreover, there was a set of 53 domains that was performing HTTP redirection without User-Agent, but JavaScript redirection when the User-Agent was set. In the later case, the HTTP response contained an HTML document similar to the one in Figure 5.12.

Malware Drops One very interesting example that shows how adversaries are hiding the behavior of a domain from automated systems and crawlers, is http://zillowhomesforsale[.]com. When no User-Agent is present, the domain always redirected to http://ww1.zillowhomesforsale[.]com/, which served us with a parking template. When the User-Agent was set, the redirection would be to either the aforementioned URL or to a completely different domain (i.e. http://rtbtracking[.]com/click?data=Mm[...]Q2&id=8c[...]d3), based on a probabilistic algorithm.

After we identified the attempt of the domains to hide their real behavior, we tried to extract further information. We setup two Virtual Machines (VMs) on a MacBook Pro running Mac OS 10.11.6 and Avast Mac Security 2015 Version 11.18 (46914) with Virus definitions version 16103000. The first VM was an Ubuntu 14.04.1 and the second a Mac OS 10.11.6. We started manually browsing to the domain names mentioned earlier and we identified several instances of malicious websites and URLs we were redirected to.

For example, zillowhomesforsale[.]com this time redirected us to http://www.searchnet[.]com/Search/Loading?v=5 which was blocked by Avast and classified as *RedirMe-inf [Trj]*, a well known trojan [4]. Similarly, when we browsed to the domain name youtubezeneletoltes[.]net we came across an automatic downloader of a disk image file named "FlashPlayer.dmg". It contained a binary that we submitted to VirusTotal for analysis. The results pointed to malware, since 15/54 Antivirus reports were suggesting some type of Trojan or Adware (`http://bit.ly/2ffwyW1`).

Domains that we visited, may also redirect to an original website (not necessarily the trademark they were abusing), after appending an affiliate identifier in the URL. For instance, visiting jcpenneyoulet[.]com lands to http://www.jcpenney[.]com/?cm_mmc=google%20non-[...] and visiting toysrusuk[.]com yields http://www.target[.]com/?clkid=4738[...].

[4] `http://malwarefixes.com/threats/htmlredirme-inf-trj/`

Social Engineering and Phishing Another type of abuse we identified was related to social engineering and phishing types of attacks. After visiting some domains like staple-seaseyrebates[.]com, we were redirected to http://viewcustomer[.]com/s3/p10/index-20 up-p10-cnf-t1-p4.php?tracker=wait.loading-links.com&keyword=staples1[...]. The landing page presented us with a survey for Staples that would reward us with a gift after completing it, clearly not related to the Staples business in any way.

We made two noteworthy observations while browsing these domains. First, the type of abuse was not related to the domain in particular, or some class of domains; every domain name that was performing redirections in this cluster was a candidate for every type of abuse mentioned so far (downloaders, drive-by, social engineering, affiliate abuse). Although it seemed that domains squatting trademarks related to e-shops (JCPenney, Toys "R" Us, etc) would perform affiliate abuse, it was not empirically found to be true. On the contrary, we were redirected to every type of abuse, from every domain we investigated, something that we expected based on our clustering technique.

Our second observation has to do with the redirection chain itself for the domain names when involved in affiliate abuse. We previously discussed how the adversaries are trying to hide the suspicious behavior of their combosquatting domains from automated headless crawlers. When manually visiting the domains however, we noticed that similar techniques are used in browsers.

CHAPTER 6

CONCLUSION

The goal of this thesis has been to show how to actively query domain names in order to assist in detecting security threats and provide context around Internet Protocol addresses. Through a novel dataset, publicly available to the scientific community and operational researchers, it attempts to bridge the gap between data and information that was becoming available through full packet inspection and the endured losses from the lack of it.

The first study presented in this thesis discussed issues revolving around IP intelligence and introduced a system capable of collecting DNS data, namely Active DNS, that can provide an adequate alternative to Passive DNS data. The system, Thales, is able to generate billiond of resolution requests on a daily basis, collect the appropriate responses, and store the data in a usable big data format for a prolonged period of time. We demonstrated how the system operates and the stability of data collected over six months. By comparing the newly collected Active DNS data to Passive DNS data, the study showed that Active DNS can provide information around malicious domains much sooner than Passive DNS, and applications built using Passive DNS data (e.g., Alembic [71], Section 3.4.2, p. 44), can operate with Active DNS in an adequate capacity.

The next study focused on architectural changes and lessons learned around the system that generates the Active DNS dataset. It discussed how Thales was upgraded over the years to Thales 2.0, which provides a much more stable stream of data, higher query volume, easier integration, horizontal scalability, and higher availability than Thales. Through a series of measurements, we showed how the new system outperforms the older one, and how the increased amount of data can further help researchers in the security community.

Finally, the last part of this thesis, showcased *Combosquatting*, a technique widely used by miscreants to hide attacks in plain sight. Utilizing the Active DNS datasets mentioned

earlier, the study showed how popular brand names and domain names are being used as part of newly registered domains, in order to trick users into thinking they are interacting with a legitimate website or service. With over two million combosquatting domain names registered over the course of approximately five years, the study shows that adversaries use combosquatting in a variety of different attacks and clients resolve those domains at a peak of billions of times per day.

In summary, this thesis introduces a novel system to collect a large DNS dataset as an alternative to the widely used Passive DNS data, that can help the community tackle emerging threats, in Chapter 3, p. 25. Then, it walks through changes to the Active DNS data collection system (Chapter 4, p. 50) and lessons learned after operating the system for almost five years. Finally, in Chapter 5, p. 108, this thesis utilizes the newly introduced Active DNS data and older Passive DNS data to study the emerging Combosquatting threats and how they are targeting unsuspected users and businesses in various different attack types, ranging from phishing and social engineering, to Advanced Persistent Threats (APT).

6.1 Considerations and Limitations

System development and Internet measurements often have certain limitations and restrictions that can affect both performance and accuracy. While such impediments do not invalidate the results of this thesis, it is important to discuss them. This sections walks through the limitations encountered in each of the three studies presented in this thesis.

6.1.1 Active DNS Limitations

The goal of this study was to build a system that can generate large DNS datasets that can be used as an alternative to the widely used Passive DNS data. To that end, a distributed querying engine was developed and the data collected was evaluated against Passive DNS data.

System Architecture

The challenges we had faced in the past and limitations around the Active DNS data generation system, Thales, are discussed in Section 4.3, p. 56. Overall, the system could endure potential data loss because of the way packets were collected off of the wire, was not able to immediately validate that a DNS response arriving was part of a DNS query that was submitted earlier, and the system could be overwhelmed by Internet Background Radiation [105, 106] at times. Moreover, the system was mainly based on LXC containers that were expected to perform resolution requests, which were hard to manage and automate tasks they performed. Finally, scaling the system outside of the datacenter we used was cumbersome, since the data collection through a network span made the topology fairly static, as discussed in Section 4.3.4, p. 62.

Active DNS Data

At the same time, the study discusses the applicability of Active DNS data in security research. Even though we demonstrate how this data can be used in certain measurements and systems (Sections 3.4.1, 3.4.2, 3.4.3, pp. 40 - 47), there are certain limitations in the data itself that should be noted. First, the Active DNS data, since it is actively generated, does not contain any client data, unlike Passive DNS, and cannot yield any behavioral information regarding clients or applications. Second, the data includes very few non-existent domain names (`NXDOMAIN`s), since Thales mostly queries for domain names that exist in zonefiles from TLDs. Third, the domains are queried periodically, at least twice a day, which means that results of domain name to RDATA mappings will only be available with approximately 12-hour intervals. Hence, if something changes on the Internet in the meantime, the data might be late to reflect that change, but will do within the same 24-hour period. Fourth, the Active DNS data does not include child labels under domain names in zonefiles. The only child labels we find in Active DNS come from the (non-negligible) lists partners share and public Open Source Intelligence (OSINT). These lists can compose

up to 40% of the Active DNS seed data, however, the domain to RDATA associations in Active DNS are much more sparse than Passive DNS. Hence, we can only query whatever we know exists and we can only have data in the Active DNS datasets that we have queried for.

6.1.2 Thales 2.0 Limitations

The redesign and new architecture of the Active DNS data collection system is expected to solve the limitations and issues that had been faced with Thales in the past. Nevertheless, the newly deployed Active DNS system, Thales 2.0, still has two limitations we should acknowledge.

First, the highly distributed nature of the system makes it significantly dependent on network connectivity and high bandwidth network connections. The current system is running on 10GbE network appliances without saturation. However, adding more nodes, distributed in different datacenters and geographic locations, will require a significantly fast network connection between devices that generate data (Step 5, Figure 4.1, p. 53) and the centralized Kafka cluster (Step 7, Figure 4.1, p. 53)).

Second, the system lacks a completely automated auditing framework. The first step in our road map for the future of Thales 2.0 is automating audit controls for every individual component on the system. Currently, the system has the ability to recover from random errors (e.g., power outages, network disconnects, fragmentation, etc.), however, reasons behind those issues need heavy manual effort to uncover. As we have seen in the past, manual intervention can be harmful for the longevity of the system, hence, automation is critical.

6.1.3 Combosquatting Limitations

The goal of the Combosquatting study is to shed light into the use of combosquatting domain names in cyber threats. To do that end, we device a methodology that will identify

combosquatting domain names, measure their prevalence in DNS data, and actively collect website data to determine the way domains are used in attacks.

Combosquatting Identification

In order to identify the way combosquatting domain names are used, we limit ourselves to 268 domain names for popular brands and trademarks. Hence, it is difficult to generalize to every trademark, since different domains might be abused in different ways, and identifying every single combosquatting domain can be hard. For instance, there are cases of domains like `comcast-universal[.]com`, which includes two trademarks, Comcast and Universal, but we do not know which one the attacker wanted to abuse. However, both Comcast and Universal are brands owned by the same company, therefore, examples like this do affect the same entity. In cases where the trademarks abused are from different entities, it might be harder to single out the entity in question, without further investigation (e.g., look at a website, if one exists, look at victims, etc).

Moreover, there are trademarks that have been excluded from our analysis due to the very generic term trademarked. For instance, as discussed in Section 5.3.1, p. 115 and Table 5.3, p. 116, the word *apple* can refer to Apple Inc. or the fruit apple. Hence, a domain name like `apple-wholesale[.]com` could be either a fake e-shop trying to lure customers into buying counterfeit products, or a wholesale business that sells apples. Therefore, in order to avoid false positives in our analysis, we have removed such words and focused only on trademarks that have a much lower false positive rate in our identification methodology.

DNS Data

The datasets used for this study come from several sources, as mentioned in Section 5.3.2, p. 117 and Table 5.4, p. 116. Our analysis related to Passive DNS data ($PDNS$) reflects data collected and reduced at the recursives of a large ISP in North America. Client in-

formation has been removed to protect users' privacy, hence our measurements are purely based on aggregate statistics. Thus, we cannot differentiate between one client making billions of resolution requests for the same domain, versus millions of clients making thousands of requests. However, since the data comes from an ISP that offers both residential and business services, we expect a query distribution that would not be biased towards a handful of clients performing an excessive amount of requests. Given that the requests we see are for many different domains, and persist across days, we expect this behavior to better fit organic, user-generated, traffic.

On the other hand, the Active DNS data ($ADNS$) is limited to domain names that the Active DNS system has collected. As mentioned earlier, the Active DNS dataset is limited to only domain names that have been found in a TLD zonefile, or were provided by an external partner. Hence, there might be many other combosquatting domain names on the Internet, which we have no visibility into. Because the overlap of the Passive and Active DNS datasets is very small (only a few days), we cannot draw a statistically significant conclusion in order to extrapolate from one vantage point to the other.

Web Crawling & Classification

The last part of our Combosquatting work attempts to identify the use of combosquatting domains on the Internet. To do that, we collect daily screenshots of the websites hosted on the domain names and then cluster them based on their visual similarity. Given that websites can change radically before an attack, when an attack takes place, and after the attack has concluded, there is a chance we may have missed an ongoing attack. Therefore, we use the term *Suspicious* in Section 5.5.1, p. 134 and Table 5.8, p. 136, as a class of domains for which we did not witness an attack, but the domain itself is not (1) owned by the trademark it abuses, nor (2) served by the same authority (or a trademark protection company, like MarkMonitor). Hence, when a combosquatting domain serves a parking page, or an under construction page, we cannot safely assume it is benign. This page might

change for a few minutes or hours to a phishing page and then back to the old parking page. If we do not crawl the website at the right time, we may never know its real nature. Thus, to avoid confusion, we label such domain names as suspicious.

6.2 Closing Remarks

This thesis showed how to actively query domain names in order to assist in detecting security threats and provide context around Internet Protocol addresses. The first study presented in this thesis provides a thorough description of a system that can actively collect DNS data, made publicly available to the research community. The data is shown to be adequate to use in security research, as an alternative to the widely used Passive DNS datasets. Active DNS data is shown to provide a much better breadth of the Internet infrastructure than Passive DNS, and does so in a timely manner. The second study provides a detailed analysis of the architectural changes the Active DNS data collection system went through over almost five years. Several problems and issues that were faced over the years mandated changes that made the system more robust and increased its availability. Moreover, these changes increased the quality and quantity of the data, both in raw numbers, and when it comes to security research. Finally, the last study, takes advantage of the Active DNS data and older Passive DNS data, to explore Combosquatting, an attack technique used for over five years, in order to hide Internet threats in plain sight. Miscreants have registered millions of combosquatting domain names, which have been resolved billions of times, and have been used in several different attack types, ranging from phishing and social engineering, to Advanced Persistent Threats (APT).

These studies provide novel datasets to the security community that can be used to replace or complement very expensive proprietary datasets, and also demonstrate the applicability of such datasets, their growth over almost five years, and how they can be used in security research. In summary, this thesis should assist network security researchers in data collection, enable research through readily available big DNS data, and demonstrate

how these datasets can be effectively used to detect abuse.

Appendices

APPENDIX A

COMBOSQUATTING

A.1 APT Domains

Table A.1 shows a list of combosquatting domain names related to Advanced Persistent Threats (APT). These domains were found in the public APT reports available at `http://tinyurl.com/apt-reports` and our CP and CA datasets (Table 5.5, p. 117).

Table A.1: Combosquatting domains related to APT.

Trademark	Domain	APT	Activity Period	Attribution	Reference
Adobe	adobearm[.]com	DarkHotel	5/12 - 11/14	Unknown Actor	[143]
Adobe	adobekr[.]com	Dust Storm	5/10 - 2/16	Unknown Actor	[144]
Adobe	adobeplugs[.]net	DarkHotel	5/12 - 11/14	Unknown Actor	[143]
Adobe	adobeservice[.]net	TooHash	Unknown - 10/14	Chinese Origin	[145]
Adobe	adobeupdates[.]com	DarkHotel	5/12 - 11/14	Unknown Actor	[143]
Adobe	adobeus[.]com	Dust Storm	5/10 - 2/16	Unknown Actor	[144]
Adobe	plugin-adobe[.]com	Saffron Rose	Unknown - 5/14	Iranian Origin	[146]
Amazon	amazonwikis[.]com	Dust Storm	5/10 - 2/16	Unknown Actor	[144]
Delta	deltae[.]com[.]br	Comment Crew	Unknown - 2/13	Unknown Actor	[147]
Delta	deltateam[.]ir	Snake/Uroboros	Uknown - 8/14	Unknown Actor	[148]
Delta	leveldelta[.]com	MiniDuke	2/13 - 5/13	Unknown Actor	[149]
Dropbox	online-dropbox[.]com	Asruex	10/15 - 6/16	Unknown Actor	[150]
Facebook	privacy-facebook[.]me	Pawn Storm	2/16 - 4/16	Unknown Actor	[151]
Facebook	users-facebook[.]com	Saffron Rose	Unknown - 5/14	Iranian Origin	[146]
Facebook	xn--facebook-06k[.]com	Saffron Rose	Unknown - 5/14	Iranian Origin	[146]

Continued on next page.

Table A.1 — Continued from previous page.

Trademark	Domain	APT	Activity Period	Attribution	Reference
Google	all-google[.]com	SpyNet	Unknown - 8/14	Unknown Actor	[152]
Google	drive-google[.]co	Rocket Kitten	Unknown - 11/15	Iranian Origin	[153]
Google	drives-google[.]co	Rocket Kitten	Unknown - 11/15	Iranian Origin	[153]
Google	google-blogspot[.]com	Quartermaster/Sunshop	5/13 - 11/13	Chinese Origin	[154]
Google	google-config[.]com	Comfoo	Unknown - 7/13	Unknown Actor	[155]
Google	google-dash[.]com	Turbo Twist	4/16 - 4/16	C0d0s0 Team	[156]
Google	google-login[.]com	Comfoo	Unknown - 7/13	Unknown Actor	[155]
Google	google-office[.]com	Enfal	Unknown - 9/11	Chinese Origin	[157]
Google	google-officeonline[.]com	Enfal	Unknown - 9/11	Chinese Origin	[157]
Google	google-setting[.]com	Rocket Kitten	Unknown - 11/15	Iranian Origin	[153]
Google	google-verify[.]com	Rocket Kitten	Unknown - 11/15	Iranian Origin	[153]
Google	googlecaches[.]com	ScanBox	9/14 - 10/14	Unknown Actor	[158]
Google	googlenewsup[.]net	Roaming Tiger	Unknown - 7/14	Chinese Origin	[159]
Google	googlesale[.]net	Ixeshe	3/14 - 6/14	Chinese Origin	[160]
Google	googlesetting[.]com	Sofacy	4/15 - 5/15	Russian Origin	[161]

Continued on next page.

Table A.1 — Continued from previous page.

Trademark	Domain	APT	Activity Period	Attribution	Reference
Google	googletranslatione[.]com	Trochilus	6/15 - 1/16	Unknown Actor	[162]
Google	googleupdate[.]hk	Comfoo	Unknown - 7/13	Unknown Actor	[155]
Google	googlewebcache[.]com	ScanBox	9/14 - 10/14	Unknown Actor	[158]
Google	imggoogle[.]com	DarkHotel	5/22 - 11/14	Unknown Actor	[143]
Google	privacy-google[.]com	Saffron Rose	Unknown - 5/14	Iranian Origin	[146]
Google	webmailgoogle[.]com	ScanBox	9/14 - 10/14	Unknown Actor	[158]
Google	xn--google-yri[.]com	Saffron Rose	Unknown - 5/14	Iranian Origin	[146]
iCloud	localiser-icloud[.]com	Pawn Storm	2/16 - 4/16	Unknown Actor	[151]
iCloud	securityicloudservice[.]com	Pawn Storm	2/16 - 4/16	Unknown Actor	[151]
Microsoft	ftpmicrosoft[.]com	Quartermaster/Sunshop	5/13 - 11/13	Chinese Origin	[154]
Microsoft	microsoft-cache[.]com	Turbo Twist	4/16 - 4/16	C0d0s0 Team	[156]
Microsoft	microsoft-security-center[.]com	Suckfly	7/15 - 5/16	Unknown Actor	[163]
Microsoft	microsoft-xpupdate[.]com	DarkHotel	5/12 - 11/14	Unknown Actor	[143]
Microsoft	microsoftc1pol361[.]com	Carbanak	10/14 - 2/15	Unknown Actor	[164]
Microsoft	microsoftmse[.]com	Four Element Sword	10/14 - 4/16	Unknown Actor	[165]

Continued on next page.

Table A.1 — Continued from previous page.

Trademark	Domain	APT	Activity Period	Attribution	Reference
Mozilla	mozillacdn[.]com	Poseidon Group	Unknown - 2/16	Poseidon Group	[165]
Reuters	reuters-press[.]com	Pawn Storm	2/16 - 4/16	Unknown Actor	[151]
Skype	downloadskype[.]cf	PoisonIvy	6/14 - 4/15	Israelian Origin	[166]
Yahoo	cc-yahoo-inc[.]org	Pawn Storm	2/16 - 4/16	Unknown Actor	[151]
Yahoo	delivery-yahoo[.]com	Sofacy II	Unknown - 4/15	Unknown Actor	[167]
Yahoo	edit-mail-yahoo[.]com	Pawn Storm	2/16 - 4/16	Unknown Actor	[151]
Yahoo	help-yahoo-service[.]com	Pawn Storm	2/16 - 4/16	Unknown Actor	[151]
Yahoo	newesyahoo[.]com	Apt Against India	Unknown - 8/13	Unknown Actor	[168]
Yahoo	privacy-yahoo[.]com	Sofacy II	Unknown - 4/15	Unknown Actor	[167]
Yahoo	settings-yahoo[.]com	Sofacy II	Unknown - 4/15	Unknown Actor	[167]
Yahoo	us-mg6mailyahoo[.]com	Strontium	Unknown - 11/15	Unknown Actor	[169]
Yahoo	yahoo-config[.]com	Comfoo	Unknown - 7/13	Unknown Actor	[155]
Yahoo	yahoo-user[.]com	Comfoo	Unknown - 7/13	Unknown Actor	[155]
Yahoo	yahooeast[.]net	Hidden Lynx	Unknown - 9/13	Unknown Actor	[170]
Yahoo	yahooip[.]net	EvilGrab	9/13 - 1/14	Unknown Actor	[171]

Continued on next page.

Table A.1 — Continued from previous page.

Trademark	Domain	APT	Activity Period	Attribution	Reference
Yahoo	yahoomail[.]com[.]co	Saffron Rose	Unknown - 5/14	Iranian Origin	[146]
Yahoo	yahooprotect[.]com	EvilGrab	9/13 - 1/14	Unknown Actor	[171]
Yahoo	yahooprotect[.]net	EvilGrab	9/13 - 1/14	Unknown Actor	[171]
Yahoo	yahooservice[.]biz	DarkHotel	5/12 - 11/14	Unknown Actor	[143]
Yahoo	yahoowebnews[.]com	IceFrog	Unknown - 9/13	Chinese Origin	[172]

REFERENCES

[1] J. Postel, *Internet Protocol*, RFC 791 (INTERNET STANDARD), Updated by RFCs 1349, 2474, 6864, Internet Engineering Task Force, Sep. 1981.

[2] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000.

[3] S. E. Smaha, "Haystack: An intrusion detection system," in *[Proceedings 1988] Fourth Aerospace Computer Security Applications*, IEEE, 1988, pp. 37–44.

[4] R. Heady, G. Luger, A. Maccabe, and M. Servilla, "The architecture of a network level intrusion detection system," Los Alamos National Lab., NM (United States); New Mexico Univ., Albuquerque …, Tech. Rep., 1990.

[5] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[6] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering*, no. 2, pp. 222–232, 1987.

[7] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks.," in *Lisa*, vol. 99, 1999, pp. 229–238.

[8] B. Claise, *Cisco Systems NetFlow Services Export Version 9*, RFC 3954 (Informational), Internet Engineering Task Force, Oct. 2004.

[9] G. Schaffrath and B. Stiller, "Conceptual Integration of Flow-Based and Packet-Based Network Intrusion Detection," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*, Springer, 2008, pp. 190–194.

[10] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An Overview of IP Flow-Based Intrusion Detection," *IEEE communications surveys & tutorials*, vol. 12, no. 3, pp. 343–356, 2010.

[11] J. Daemen and V. Rijmen, "Reijndael: The Advanced Encryption Standard.," *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 26, no. 3, pp. 137–139, 2001.

[12] Zaw, Tin and Morrison, Reed and Peters, Robert J, *END-TO-END CERTIFICATE PINNING*, US Patent 9,847,992, Dec. 2017.

[13] P. Mockapetris, *Domain names: Concepts and facilities*, RFC 882, Obsoleted by RFCs 1034, 1035, updated by RFC 973, Internet Engineering Task Force, Nov. 1983.

[14] ——, *Domain names: Implementation specification*, RFC 883, Obsoleted by RFCs 1034, 1035, updated by RFC 973, Internet Engineering Task Force, Nov. 1983.

[15] C. Lever, P. Kotzias, D. Balzarotti, J. Caballero, and M. Antonakakis, "A Lustrum of Malware Network Communication: Evolution and Insights," in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017, pp. 788–804.

[16] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a Dynamic Reputation System for DNS," in *the Proceedings of 19th USENIX Security Symposium (USENIX Security '10)*, 2010.

[17] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon, "Detecting Malware Domains in the Upper DNS Hierarchy," in *the Proceedings of 20th USENIX Security Symposium (USENIX Security '11)*, 2011.

[18] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware," in *the Proceedings of 21th USENIX Security Symposium (USENIX Security '12)*, 2012.

[19] Y. Chen and M. Antonakakis and R. Perdisci and Y. Nadji and D. Dagon and W. Lee, "DNS Noise: Measuring the Pervasiveness of Disposable Domains in Modern DNS Traffic," in *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, Jun. 2014, pp. 598–609.

[20] Bilge, Leyla and Balzarotti, Davide and Robertson, William and Kirda, Engin and Kruegel, Christopher, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ACM, 2012, pp. 129–138.

[21] B. Zdrnja, N. Brownlee, and D. Wessels, "Passive Monitoring of DNS Anomalies," in *Proceedings of DIMVA Conference*, 2007.

[22] R. Perdisci, I. Corona, D. Dagon, and W. Lee, "Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces," in *Proceedings of ACSAC*, Honolulu, Hawaii, USA, 2009.

[23] Lever, Chaz and Walls, Robert and Nadji, Yacin and Dagon, David and McDaniel, Patrick and Antonakakis, Manos, "Domain-Z: 28 Registrations Later," 2016.

[24] Chen, Yizheng and Kintis, Panagiotis and Antonakakis, Manos and Nadji, Yacin and Dagon, David and Lee, Wenke and Farrell, Michael, "Financial Lower Bounds of Online Advertising Abuse," in *Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment-Volume 9721*, Springer-Verlag New York, Inc., 2016, pp. 231–254.

[25] Zakir Durumeric and David Adrian and Ariana Mirian and Michael Bailey and J. Alex Halderman, "A Search Engine Backed by Internet-Wide Scanning," in *Proceedings of the 22nd ACM Conference on Computer and Communications Security*, Oct. 2015.

[26] Perdisci, Roberto and Corona, Igino and Giacinto, Giorgio, "Early Detection of Malicious Flux Networks via Large-Scale Passive Dns Traffic Analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 5, pp. 714–726, 2012.

[27] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the Mirai Botnet," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1093–1110.

[28] C. Lever, M. Antonakakis, B. Reaves, P. Traynor, and W. Lee, "The Core of the Matter: Analyzing Malicious Traffic in Cellular Carriers," in *NDSS*, 2013.

[29] Y. Nadji, M. Antonakakis, R. Perdisci, D. Dagon, and W. Lee, "Beheading Hydras: Performing Effective Botnet Takedowns," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 121–132.

[30] P. Mockapetris, *Domain names - concepts and facilities*, RFC 1034 (INTERNET STANDARD), Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936, Internet Engineering Task Force, Nov. 1987.

[31] ——, *Domain names - implementation and specification*, RFC 1035 (INTERNET STANDARD), Internet Engineering Task Force, Nov. 1987.

[32] Son, Sooel and Shmatikov, Vitaly, "The Hitchhiker's Guide to DNS Cache Poisoning," in *International Conference on Security and Privacy in Communication Systems*, Springer, 2010, pp. 466–483.

[33] Klein, Amit, "BIND 9 DNS Cache Poisoning," *Report, Trusteer, Ltd*, vol. 3, 2007.

[34] Davies, Kim, *DNS Cache Poisoning Vulnerability Explanation and Remedies*, 2008.

[35] Moon, David A, "Chaosnet," 1981.

[36] Champine, George A, *MIT Project Athena*. Elsevier, 1991.

[37] Kintis, Panagiotis and Dagon, David and Antonakakis, Manos, "Authority Robustness Scoring Via Glue Cycle Detection," in *DNS and Internet Naming Research Directions*, Nov. 2016.

[38] B. Laurie, G. Sisson, R. Arends, and D. Blacka, *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*, RFC 5155 (Proposed Standard), Updated by RFCs 6840, 6944, Internet Engineering Task Force, Mar. 2008.

[39] F. Weimer, "Passive DNS replication," in *Proceedings of FIRST Conference on Computer Security Incident*, Hand ling, Singapore, 2005.

[40] M. Antonakakis, D. Dagon, X. Luo, R. Perdisci, W. Lee, and J. Bellmor, "A Centralized Monitoring Infrastructure for Improving DNS Security," in *Recent Advances in Intrusion Detection*, Springer, 2010, pp. 18–37.

[41] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "EXPOSURE: Finding malicious domains using passive dns analysis," in *Proceedings of NDSS*, 2011.

[42] D. Plonka and P. Barford, "Context-aware Clustering of DNS Query Traffic," in *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '08, Vouliagmeni, Greece: ACM, 2008, pp. 217–230, ISBN: 978-1-60558-334-1.

[43] Ma, Justin and Saul, Lawrence K and Savage, Stefan and Voelker, Geoffrey M, "Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Jun. 2009.

[44] B. Rahbarinia, R. Perdisci, and M. Antonakakis, "Segugio: Efficient Behavior-Based Tracking of Malware-Control Domains in Large ISP Networks," in *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*, Jun. 2015, pp. 403–414.

[45] S. Krishnan and F. Monrose, "An empirical study of the performance, security and privacy implications of domain name prefetching," in *Dependable Systems Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, Jun. 2011, pp. 61–72.

[46] Hao, Shuang and Kantchelian, Alex and Miller, Brad and Paxson, Vern and Feamster, Nick, "PREDATOR: Proactive Recognition and Elimination of Domain Abuse at Time-Of-Registration," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2016, pp. 1568–1579.

[47] Liu, Daiping and Hao, Shuai and Wang, Haining, "All Your DNS Records Point to Us: Understanding the Security Threats of Dangling DNS Records," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2016, pp. 1414–1425.

[48] Jakobsson, Markus and Tsow, Alex and Shah, Ankur and Blevis, Eli and Lim, Youn-Kyung, "What instills trust? a qualitative study of phishing," in *Financial Cryptography and Data Security*, Springer, 2007, pp. 356–361.

[49] Jakobsson, Markus, "The human factor in phishing," *Privacy & Security of Consumer Information*, vol. 7, no. 1, pp. 1–19, 2007.

[50] Wang, Yi-Min and Beck, Doug and Wang, Jeffrey and Verbowski, Chad and Daniels, Brad, "Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting.," *SRUTI*, vol. 6, pp. 31–36, 2006.

[51] Agten, Pieter and Joosen, Wouter and Piessens, Frank and Nikiforakis, Nick, "Seven months' worth of mistakes: A longitudinal study of typosquatting abuse," in *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)*, Internet Society, 2015.

[52] Nikiforakis, Nick and Van Acker, Steven and Meert, Wannes and Desmet, Lieven and Piessens, Frank and Joosen, Wouter, "Bitsquatting: Exploiting bit-flips for fun, or profit?" In *Proceedings of the 22nd international conference on World Wide Web*, ACM, 2013, pp. 989–998.

[53] A. Dinaburg, "Bitsquatting: DNS Hijacking without Exploitation," in *Proceedings of BlackHat Security*, Jul. 2011.

[54] Nikiforakis, Nick and Balduzzi, Marco and Desmet, Lieven and Piessens, Frank and Joosen, Wouter, "Soundsquatting: Uncovering the use of homophones in domain squatting," in *Information Security*, Springer, 2014, pp. 291–308.

[55] "Combosquatting: The Business of Cybersquatting," in *FairWinds Partners, LLC*, 2008.

[56] *LinuxContainers.org*, `https://linuxcontainers.org/`, 2016.

[57] Alexa, *The Web Information Company*, `http://www.alexa.com/`, 2016.

[58] *Common Crawl*, https://commoncrawl.org/, 2016.

[59] *Domain Blacklist: abuse.ch*, http://www.abuse.ch/, 2016.

[60] *Malware Domain List*, http://www.malwaredomainlist.com/forums/index.php?topic=3270.0, 2016.

[61] *Domain Blacklist: Blackhole DNS*, http://www.malwaredomains.com/wordpress/?page_id=6, 2016.

[62] *Domain Blacklist: sagadc*, http://dns-bh.sagadc.org/, 2016.

[63] *Domain Blacklist: hphosts*, http://hosts-file.net/?s=Download, 2016.

[64] *Domain Blacklist: SANS*, https://isc.sans.edu/suspicious_domains.html, 2016.

[65] *Domain Blacklist: itmate*, http://vurl.mysteryfcm.co.uk/, 2016.

[66] Y. Nadji, M. Antonakakis, R. Perdisci, and W. Lee, "Connected Colors: Unveiling the Structure of Criminal Networks," in *Research in Attacks, Intrusions, and Defenses*, Springer, 2013, pp. 390–410.

[67] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and Detecting Fast-Flux Service Networks," in *NDSS*, 2008.

[68] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "Phishnet: Predictive blacklisting to detect phishing attacks," in *INFOCOM, 2010 Proceedings IEEE*, IEEE, 2010, pp. 1–5.

[69] K. Ishibashi, T. Toyono, H. Hasegawa, and H. Yoshino, "Extending black domain name list by using co-occurrence relation between dns queries," *IEICE transactions on communications*, vol. 95, no. 3, pp. 794–802, 2012.

[70] M. Felegyhazi, C. Kreibich, and V. Paxson, "On the Potential of Proactive Domain Blacklisting," in *Proceedings of the 3rd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More (LEET)*, Apr. 2010.

[71] C. Lever, R. Walls, Y. Nadji, D. Dagon, P. McDaniel, and M. Antonakakis, "Domain-Z: 28 Registrations Later Measuring the Exploitation of Residual Trust in Domains," in *37th IEEE International Symposium on Security and privacy*, May 2016.

[72] L. Daigle, *WHOIS Protocol Specification*, RFC 3912 (Draft Standard), Internet Engineering Task Force, Sep. 2004.

[73] *Domain Graveyard*, http://domaingraveyard.com/, 2016.

[74] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, *Address Allocation for Private Internets*, RFC 1918 (Best Current Practice), Updated by RFC 6761, Internet Engineering Task Force, Feb. 1996.

[75] M. Cotton and L. Vegoda, *Special Use IPv4 Addresses*, RFC 5735 (Best Current Practice), Obsoleted by RFC 6890, updated by RFC 6598, Internet Engineering Task Force, Jan. 2010.

[76] J. Weil, V. Kuarsingh, C. Donley, C. Liljenstolpe, and M. Azinger, *IANA-Reserved IPv4 Prefix for Shared Address Space*, RFC 6598 (Best Current Practice), Internet Engineering Task Force, Apr. 2012.

[77] B. Coat, *Snake In The Grass: Python-based Malware Used For Targeted Attacks*, https://www2.bluecoat.com/security-blog/2014-06-10/snake-grass-python-based-malware-used-targeted-attacks, 2014.

[78] M. L. C. C. Security, *CopyKittens Attack Group*, https://eforensicsmag.com/copykittens/, 2015.

[79] Mandiant, "APT1," Tech. Rep., 2013, http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf.

[80] P. Kintis, N. Miramirkhani, C. Lever, Y. Chen, R. Romero-Gómez, N. Pitropakis, N. Nikiforakis, and M. Antonakakis, "Hiding in Plain Sight: A Longitudinal Study of Combosquatting Abuse," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 569–586.

[81] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "A Survey on Malicious Domains Detection Through DNS Data Analysis," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[82] K. Tian, S. T. Jan, H. Hu, D. Yao, and G. Wang, "Needle in a Haystack: Tracking Down Elite Phishing Domains in the Wild," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 429–442.

[83] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, and A. Mosquera, "Detecting DGA Domains With Recurrent Neural Networks and Side Information," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, 2019, pp. 1–10.

[84] X. Mi, X. Feng, X. Liao, B. Liu, X. Wang, F. Qian, Z. Li, S. Alrwais, L. Sun, and Y. Liu, "Resident Evil: Understanding Residential IP Proxy as a Dark Service," in *2019 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2019, pp. 1185–1201.

[85] Athanasios Kountouras and Panagiotis Kintis and Chaz Lever and Yizheng Chen and Yacin Nadji and David Dagon and Manos Antonakakis and Rodney Joffe, "Enabling Network Security Through Active DNS Datasets," in *Research in Attacks, Intrusions, and Defenses - 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings*, 2016, pp. 188–208.

[86] CZDS, *Centralized Zone Data Service (CZDS)*, 2020.

[87] MAXMIND, *Messaging that just works*, 2020.

[88] M. Gieben, *Alternative (more granular) approach to a DNS library*, 2020.

[89] ZMAP, *ZDNS*, 2020.

[90] *Docker - Build, Ship and Run, Any App, Anywhere*, `https://www.docker.com`, 2015.

[91] Apache, *Apache Avro*, 2020.

[92] Confluent, *Confluent: Apache Kafka & Event Streaming Platform for the Enterprise*, 2020.

[93] ——, *Capacity planning and sizing — Confluent Platform*, 2020.

[94] J. Kreps, *Benchmarking Apache Kafka: 2 Million Writes Per Second (On Three Cheap Machines)*, 2014.

[95] T. Palino, *Running Kafka At Scale*, 2015.

[96] T. Mancill, *20 Best Practices for Working With Apache Kafka at Scale*, 2018.

[97] Apache, *Apache Spark — Unified Analytics Engine for Big Data*, 2020.

[98] ——, *Apache Parquet*, 2020.

[99] RIPE-NCC, *Hadoop PCAP library*, 2020.

[100] HashiCorp, *Nomad by HashiCorp*, 2020.

[101] Apache, *Apache Flume*, 2020.

[102] ——, *Apache NiFi*, 2020.

[103] ——, *Apache Gobblin*, 2020.

[104] N. Labs, *NLnet Labs — Unbound*, 2020.

[105] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of Internet Background Radiation," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, 2004, pp. 27–40.

[106] E. Wustrow, M. Karir, M. Bailey, F. Jahanian, and G. Huston, "Internet Background Radiation Revisited," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, 2010, pp. 62–74.

[107] T. Cymru, *The Bogon Reference*, 2020.

[108] Tanenbaum, Andrew S and Van Steen, Maarten, *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[109] J. Damas, M. Graff, and P. Vixie, *Extension Mechanisms for DNS (EDNS(0))*, RFC 6891 (INTERNET STANDARD), Internet Engineering Task Force, Apr. 2013.

[110] Anagnostopoulos, Marios and Kambourakis, Georgios and Kopanos, Panagiotis and Louloudakis, Georgios and Gritzalis, Stefanos, "DNS Amplification Attack Revisited," *Computers & Security*, vol. 39, pp. 475–485, 2013.

[111] C. Contavalli, W. Van Der Gaast, D Lawrence, and W. Kumari, *Client Subnet in DNS Queries*, RFC 7871 (Informational), Internet Engineering Task Force, May 2016.

[112] D. N. Stat, *Domain name registration's statistics*, 2019.

[113] VERISIGN, *VERISIGN Q2 2019 DOMAIN NAME INDUSTRY BRIEF: INTERNET GROWS TO 354.7 MILLION DOMAIN NAME REGISTRATIONS IN THE SECOND QUARTER OF 2019*, 2019.

[114] A. Portier, H. Carter, and C. Lever, "Security in Plain TXT," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Springer, 2019, pp. 374–395.

[115] Durumeric, Zakir and Wustrow, Eric and Halderman, J Alex, "ZMap: Fast Internet-wide scanning and its security applications," in *22nd {USENIX} Security Symposium ({USENIX} Security 13)*, 2013, pp. 605–620.

[116] Vixie, P and Schryver, V, "DNS Response Policy Zones (RPZ)," *draft-vixie-dns-rpz-04 (work in progress)*, 2016.

[117] Aitchison, Ron, *Pro Dns and BIND 10*. Apress, 2011.

[118] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.

[119] *Anticybersquatting Consumer Protection Act (ACPA)*, `http://www.patents.com/acpa.htm`, Nov. 1999.

[120] Edelman, Benjamin, "Large-scale registration of domains with typographical errors," *Harvard University*, 2003.

[121] Khan, Mohammad Taha and Huo, Xiang and Li, Zhou and Kanich, Chris, "Every Second Counts: Quantifying the Negative Externalities of Cybercrime via Typosquatting," in *Proceedings of the 36th IEEE Symposium on Security and Privacy*, 2015.

[122] Janos Szurdi and Balazs Kocso and Gabor Cseh and Jonathan Spring and Mark Felegyhazi and Chris Kanich, "The Long "Taile" of Typosquatting Domain Names," in *23rd USENIX Security Symposium (USENIX Security 14)*, San Diego, CA: USENIX Association, 2014, pp. 191–206, ISBN: 978-1-931971-15-7.

[123] T. Moore and B. Edelman, "Measuring the perpetrators and funders of typosquatting," in *Financial Cryptography and Data Security*, vol. 6052, 2010, pp. 175–191.

[124] N. Nikiforakis, S. V. Acker, W. Meert, L. Desmet, F. Piessens, and W. Joosen, "Bitsquatting: Exploiting bit-flips for fun, or profit?" In *WWW'13*, 2013, pp. 989–998.

[125] Holgers, Tobias and Watson, David E. and Gribble, Steven D., "Cutting through the confusion: a measurement study of homograph attacks," in *Proceedings of the 2006 USENIX Annual Technical Conference*, Boston, MA, 2006.

[126] E. Gabrilovich and A. Gontmakher, "The homograph attack," *Communucations of the ACM*, vol. 45, no. 2, p. 128, Feb. 2002.

[127] TrendMicro, *Securing Your Journey to the Cloud*, `http://www.trendmicro.com/`, 2016.

[128] dmoz, *DMOZ - the Open Directory Project*, `http://www.dmoz.org`, 2016.

[129] *Domain Blacklist: driveby*, `http://www.blade-defender.org/eval-lab/`, 2015.

[130] Kreibich, Christian and Kanich, Chris and Levchenko, Kirill and Enright, Brandon and Voelker, Geoffrey M and Paxson, Vern and Savage, Stefan, "On the Spam Campaign Trail.," *LEET*, vol. 8, no. 2008, pp. 1–9, 2008.

[131] *Certificate transparency*, `https://www.certificate-transparency.org`, 2017.

[132] Let's Encrypt, *Let's Encrypt — Free SSL/TLS Certificates*, `https://letsencrypt.org`, 2017.

[133] J. Aas, *Let's Encrypt: The CA's Role in Fighting Phishing and Malware*, `https://letsencrypt.org/2015/10/29/phishing-and-malware.html`, 2015.

[134] Segaran, Toby and Hammerbacher, Jeff, *Beautiful data: the stories behind elegant data solutions*. " O'Reilly Media, Inc.", 2009.

[135] Ryan Kelly, *PyEnchant a spellchecking library for Python*, `http://pythonhosted.org/pyenchant/`, 2016.

[136] AllSlang, *Swear Word List & Curse Filter*, `http://www.noswearing.com/dictionary`, 2016.

[137] SOWPODS, *SOWPODS Scrabble Word List*, `https://www.wordgamedictionary.com/sowpods/`, 2016.

[138] AllSlang, *Slang Dictionary - Text Slang & Internet Slang Words*, `http://www.noslang.com/dictionary/`, 2016.

[139] Snyder, Peter and Kanich, Chris, "No please, after you: Detecting fraud in affiliate marketing networks," in *Proceedings of the Workshop on the Economics of Information Security (WEIS)*, 2015.

[140] Miramirkhani, Najmeh and Starov, Oleksii and Nikiforakis, Nick, "Dial One for Scam: A Large-Scale Analysis of Technical Support Scams," in *Proceedings of the 24th Network and Distributed System Security Symposium (NDSS 2017)*, Internet Society, 2017.

[141] Schütze, Hinrich, "Introduction to Information Retrieval," in *Proceedings of the international communication of association for computing machinery conference*, 2008.

[142] Wang, Weixin and Wang, Hui and Dai, Guozhong and Wang, Hongan, "Visualization of Large Hierarchical Data by Circle Packing," in *Proceedings of the SIGCHI*

Conference on Human Factors in Computing Systems, ser. CHI '06, Montréal, Québec, Canada: ACM, 2006, pp. 517–520, ISBN: 1-59593-372-7.

[143] Kaspersky Lab, *DARKHOTEL INDICATORS OF COMPROMISE*, https://securelist.com/files/2014/11/darkhotelappendixindicators_kl.pdf, Nov. 2014.

[144] Cylance, *OPERATION DUST STORM*, https://www.cylance.com/hubfs/2015_cylance_website/assets/operation-dust-storm/Op_Dust_Storm_Report.pdf?t=1477417126448, Feb. 2016.

[145] G DATA, *OPERATION "TOOHASH" HOW TARGETED ATTACKS WORK*, https://public.gdatasoftware.com/Presse/Publikationen/Whitepaper/EN/GDATA_TooHash_CaseStudy_102014_EN_v1.pdf, Oct. 2014.

[146] FireEye, *OPERATION SAFFRON ROSE*, https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-operation-saffron-rose.pdf, May 2013.

[147] Symantec, *Comment Crew: Indicators of Compromise*, https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/comment_crew_indicators_of_compromise.pdf, Feb. 2013.

[148] Kaspersky Lab, *The Epic Turla Operation: Solving some of the mysteries of Snake/Uroboros*, https://cdn.securelist.com/files/2014/08/KL_Epic_Turla_Technical_Appendix_20140806.pdf, Aug. 2014.

[149] Bitdefender, *A Closer Look at MiniDuke*, https://labs.bitdefender.com/wp-content/uploads/downloads/2013/04/MiniDuke_Paper_Final.pdf, May 2013.

[150] JPCERT/CC, *Asruex: Malware Infecting through Shortcut Files*, http://blog.jpcert.or.jp/2016/06/asruex-malware-infecting-through-shortcut-files.html, Jun. 2016.

[151] TrendMicro, *Looking Into a Cyber-Attack Facilitator in the Netherlands*, http://documents.trendmicro.com/assets/appendix_looking-into-a-cyber-attack-facilitator-in-the-netherlands.pdf, Apr. 2016.

[152] Marczak, William R and Scott-Railton, John and Marquis-Boire, Morgan and Paxson, Vern, "When governments hack opponents: A look at actors and technology," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 511–525.

[153] CHECK POINT SOFTWARE TECHNOLOGIES, *ROCKET KIT TEN: A CAMPAIGN WITH 9 LIVES*, http://blog.checkpoint.com/wp-content/uploads/2015/11/rocket-kitten-report.pdf, Nov. 2015.

[154] FireEye, *SUPPLY CHAIN ANALYSIS: From Quartermaster to SunshopFireEye*, https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-malware-supply-chain.pdf, Nov. 2013.

[155] SecureWorks, *Secrets of the Comfoo Masters*, https://www.secureworks.com/research/secrets-of-the-comfoo-masters, Jul. 2013.

[156] Fidelis Threat Research Team, *Turbo Twist: Two 64-bit Derusbi Strains Converge*, http://www.threatgeek.com/2016/05/turbo-twist-two-64-bit-derusbi-strains-converge.html, May 2016.

[157] TrendMicro, *THE "LURID" DOWNLOADER*, http://la.trendmicro.com/media/misc/lurid-downloader-enfal-report-en.pdf, Sep. 2011.

[158] pwc, *ScanBox framework — who's affected, and who's using it?* http://pwc.blogs.com/cyber_security_updates/2014/10/scanbox-framework-whos-affected-and-whos-using-it-1.html, Oct. 2014.

[159] Anton Cherepanov, *ScanBox framework — who's affected, and who's using it?* http://2014.zeronights.org/assets/files/slides/roaming_tiger_zeronights_2014.pdf, Jul. 2014.

[160] Asert, *Illuminating the Etumbot APT Backdoor*, https://github.com/kbandla/APTnotes/blob/master/2014/ASERT-Threat-Intelligence-Brief-2014-07-Illuminating-Etumbot-APT.pdf, Jun. 2014.

[161] root9B, *APT28 targets Financial Markets ROOT9B RELEASES ZERO DAY HASHES*, https://www.root9b.com/sites/default/files/whitepapers/R9b_FSOFACY_0.pdf, May 2015.

[162] Asert, *Uncovering the Seven Pointed Dagger Discovery of the Trochilus RAT and Other Targeted Threats*, https://goo.gl/zMbqpA, Jan. 2016.

[163] Symantec, *A Closer Look at MiniDuke*, https://www.symantec.com/connect/blogs/indian-organizations-targeted-suckfly-attacks, May 2016.

[164] Kaspersky, *CARBANAK APT THE GREAT BANK ROBBERY*, https://securelist.com/files/2015/02/Carbanak_APT_eng.pdf, Feb. 2015.

[165] Asert, *The Four Element Sword Engagement*, https://www.arbornetworks.com/blog/asert/four-element-sword-engagement/, Apr. 2016.

[166] pwc, *Attacks against Israeli & Palestinian interests*, http://pwc.blogs.com/cyber_security_updates/2015/04/attacks-against-israeli-palestinian-interests.html, Apr. 2015.

[167] ——, *Cyber Threat Operations Sofacy II– Same Sofacy, Different Day*, http://pwc.blogs.com/files/cto-tib-20150420-01a.pdf, Apr. 2015.

[168] INFOSEC CONSORTIUM, *Inside Report – APT Attacks on Indian Cyber Space*, http://ver007.com/tools/APTnotes/2013/Inside_Report_by_Infosec_Consortium.pdf, Aug. 2013.

[169] Microsoft, *Microsoft Security Intelligence Report Volume 19 — January through June, 2015*, http://download.microsoft.com/download/4/4/C/44CDEF0E-7924-4787-A56A-16261691ACE3/Microsoft_Security_Intelligence_Report_Volume_19_English.pdf, Jun. 2015.

[170] Symantec, *Hidden Lynx – Professional Hackers for Hire*, http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/hidden_lynx.pdf, Sep. 2013.

[171] TrendMicro, *2Q Report on Targeted Attack Campaigns*, http://la.trendmicro.com/media/misc/lurid-downloader-enfal-report-en.pdf, Jan. 2014.

[172] Kaspersky, *THE 'ICEFOG' APT: A TALE OF CLOAK AND THREE DAGGERS*, https://kasperskycontenthub.com/wp-content/uploads/sites/43/vlpdfs/icefog.pdf, Sep. 2013.